INTELLIGENT DATA MINING VIA EVOLUTIONARY COMPUTING

YU QI
(B. Eng, Zhejiang University)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE
2003

Acknowledgements

I would like to express my most sincere appreciation to my supervisor, Dr. K. C. Tan, for his good guidance, support and encouragement. His stimulating advice benefits me in overcoming obstacle on my research path.

Thanks to my laboratory-mate Heng Chun Meng, who has made contributions in various ways to my research work.

I am also grateful to all the individuals in Centre for Intelligent Control (CIC), as well as the technologies in the Control and Simulation Lab, Department of Electrical and Computer Engineering, National University of Singapore, which provides the research facilities to conduct the research work.

Finally, I wish to acknowledge National University of Singapore (NUS) for the financial support provided throughout my research work.

Table of Contents

List of Figures

List of Tables

Summary

This work seeks to explore the evolutionary techniques for extracting comprehensible classification rules in data mining as well as to improve its processing efficiency. For this purpose, the thesis is organised as follow:

Chapter 2 reviews the basic concept of rule induction and provides a survey on various evolutionary methods for extracting classification rules. Besides, a preliminary knowledge of coevolution and how it can be used for rule induction is also studied and discussed.

Chapter 3 presents a the two-phase approach to extract classification rules, in which a hybrid evolutionary algorithm is utilized in the first phase to confine the search space by evolving a pool of good candidate rules, e.g., genetic programming is applied to evolve nominal attributes for free structured rules and genetic algorithm is used to optimize the numeric attributes for concise classification rules without the need of discretization. These candidate rules are then used in the second phase to optimize the order and number of rules in the evolution for forming accurate and comprehensible rule sets. Good simulation results on three medical datasets show that the algorithms can be used as an assistant tool in clinical practice for better understanding and prevention of unwanted medical events.

Chapter 4 presents a distributed coevolutionary classification system (DCDM) for rule induction, which allows different species to be evolved cooperatively and simultaneously, while the computational workload is shared among multiple

computers over the Internet. Through the inter-communications among different species of rules and rule sets in a distributed computing approach, the concurrent processing and computational speed of the coevolutionary classifier are enhanced significantly. The advantages and performance of the proposed DCDM are extensively validated upon various datasets obtained from UCI machine learning repository. It is shown that the predicting accuracy of the DCDM is robust and the computational time is substantially reduced as the number of remote engines increases. Comparison results illustrate that the DCDM produces comprehensible and good classification rules for all the datasets, which are very competitive as compared with existing classifiers in literature.

Chapter 5 draws the conclusions and directions for future works.

Chapter 1

Introduction

1.1. Data mining

Advancement in the application of information technology in various fields of science, economics and industries has generated huge amounts of raw data that are beyond the processing capability of the human mind. These data, however, possibly present value in the form of knowledge that can not only provide a better understanding of the phenomenon underlying them, but also help in decision-making process.

Data mining is the automated process of discovering knowledge or information from data sources. The main challenge of data mining is to extract knowledge that is accurate, comprehensible and interesting, in spite of huge amounts of data involved and possibly noisy and unfavorable data representation. Recent developments in data mining techniques have proven its potential as a tool in the knowledge discovery process. According to Fayyad (1997), the process of data mining is just one of the steps in the overall process of discovering knowledge from data, called Knowledge Discovery from Database (KDD). This is shown in Figure 1.1.

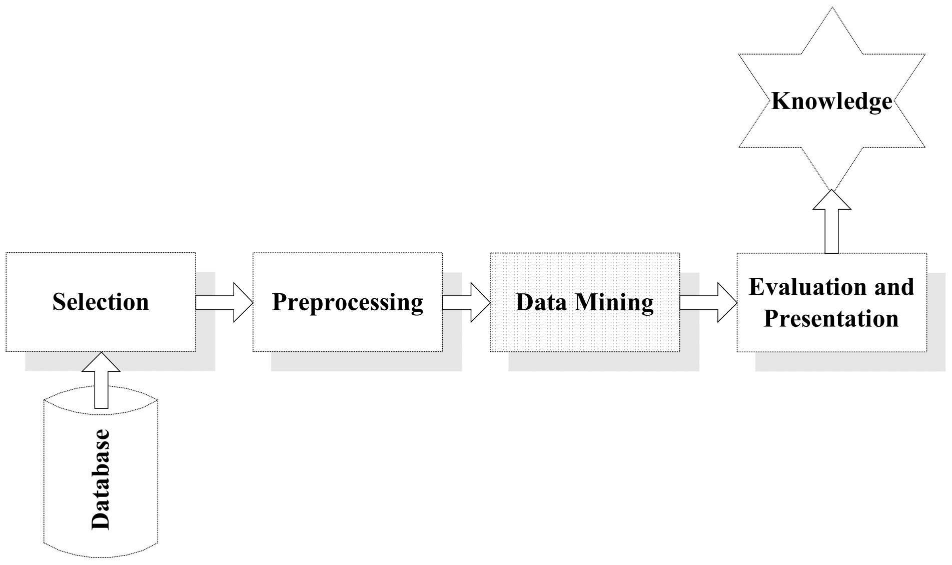


Figure 1.1: The Knowledge Discovery from Database Process

There are generally four phases in the KDD process. Firstly selection deals with identifying and understanding the goals of the entire process and collecting any prior knowledge about the data itself. The preprocessing phase cleans up the raw data, handling missing values and dealing with any data misrepresentation. Next data mining is used to extract the useful knowledge from the preprocessed data and finally, the knowledge extracted from data mining is evaluated and interpreted in the post-processing stage before it finally becomes useful knowledge. As it can be seen, data mining is a crucial phase in KDD where the knowledge is actually churned out from the data.

Knowledge discovered from data mining has many uses from classification, estimation, prediction and description to clustering. One of the most useful ways of representing the discovered knowledge is in the form of rules, with every rule representing a piece of information or knowledge. With a set of rules as knowledge about the data, tasks such as decision-making and understanding the data can be more easily accomplished. Rules have an added advantage over traditional forms of knowledge representation such as trees as more rules can be added to an existing set of rules without affecting those that are already there. Hence it can be seen that rules are a powerful and useful form of representing knowledge.

1.2. Evolutionary algorithm

Evolutionary algorithm (EA) (Michalewicz, 1994) is generally considered to consist of 4 main branches, i.e., Evolutionary Strategies (ES), Evolutionary Programming (EP), Genetic Algorithms (GA), and Genetic Programming (GP). They have been developed upon the synthesis of natural evolution, which exhibits global search capabilities by simultaneously evaluating performances at multiple points in the solution space. Before this simulated evolution process begins, an initial population of multiple coded chromosomes representing random candidate solutions is formed, and every such chromosome is assigned a performance index. At each generation of search, multiple candidates are evaluated and the search will be directed intelligently according to the Darwin's "survival-of-the-fittest" principle. Then useful search information and co-ordinates are exchanged and altered for the next generation of candidate solutions. This evolution cycle will be repeated until the final generation is reached or the solution has been found. Obviously, the computation effort involved in such an evolutionary process is massive due to the inherent nature of parallel search, particularly for complex optimization problems where a sufficiently large population and generation size that incurs a high computational workload are often needed in order to find the global optimal solutions.

1.3. Evolutionary algorithms in data mining

Among the several branches in the data mining domain, one area gaining significance is classification (Duda *et al*, 2001), which is ordinarily categorized into two groups, i.e., non-rule based and rule-based classifications. Support vector machine (Vapnik, 1995), artificial neural network (Yao and Liu, 1997), and linear genetic programming (Brameier and Banzhaf, 2001) are some of the popular approaches in the category of non-rule based classifiers. Although these methods often achieve good classification accuracy, they are generally not competent classifiers in terms of comprehensibility. However, some efforts have also been paid recently to extract explicit knowledge from the resulting model from the above methods among which, rule extraction supported NNs (Setiono, 1996; 2000) is a famous example. On the other hand, rule-based classification approaches refer to methods where explicit knowledge is derived directly from the training data and the users are able to assess the correctness and usefulness of the discovered knowledge. Comprehensible knowledge is very important in many applications, such as medical diagnosis and management decision support, where human experts play an important role in solving the problems. C4.5 (Quinlan, 1993), decision table (Kohavi, 1995), genetic algorithm-program GA-P (Howard and D'Angelo, 1995), grammar-based genetic programming (GGP) (Wong and Leung, 2000) are some of the methods in this group of classifiers.

Emulating the Darwinian-Wallace principle in natural selection and genetics, evolutionary algorithms have emerged as a promising tool for solving knowledge extraction problems in data mining. Except for the above mentioned algorithms for the classification problems, in recent years, there have been many other attempts to apply evolutionary algorithms in data mining to accomplish various tasks (Banzhaf *et el.*, 1998; Brameier and Banzhaf, 2001; Cattral, 1999; Polo and Hasse, 2000). Unlike traditional gradient-guided data mining techniques, an evolutionary algorithm intelligently searches the solution space by evaluating performances of multiple candidate solutions simultaneously and approaches the global optimum in a non-deterministic manner. Although EAs play an important role in rather widely areas of data mining domain, they have achieved more popularity in the area of rule based classification (rule induction), which might be due to the reason that sets of IF-THEN rules can easily be represented by choosing an encoding of rules that allocates specific substrings for each rule precondition and postcondtion (Mitchell, 1997). Wong and Leung (2000) proposed a grammar-based GP for the construction of classification rules. For each new problem, a domain specific grammar is defined so that the rules thus generated are more relevant and crucial to the problem. To address the issue of comprehensibility of classification rules, Bojarczuk *et al.*, (2000) implemented a non-standard tree structure GP. In this approach, the numeric attributes are discretized into nominal boundaries *a-priori* in order to use the Boolean attributes. Other EA approaches for generating classification rules in data mining include Wang *et al.*, (1998), Congdon, (2000), and Fidelis *et al.*, (2000).

However, although evolutionary algorithm is a powerful tool, the computational cost involved in terms of time and hardware increases as the size and complexity of the problem increases since it needs to perform a large number of function evaluations in parallel along the evolution process. Moreover, EA usually requires a large population and generation size in order to simulate a more realistic evolutionary model with a better approximation and resolution, which is sometimes cost prohibitive or cannot be performed without the help of high performance computing. To reduce the complexity of the solution space and make the good solutions easier to be found, the multi-stage scheme is utilized by many classification systems. Marmelstein *et al* (1998), in their Genetic Rule and Classifier Construction Environment (GRaCCE), introduced a multi-stage means for extracting classification rules from data. Genetic Algorithm (GA) based approach is to first reduce the feature set and then locate class homogeneous regions within the data. Classification rules are subsequently generated from these regions. So in the early stages of these approaches, feature extraction or dimension reduction of the dataset is the major task and classifiers or rules will not be constructed until the later stages. Kim and Han (2000), and Liu *et al.*, (2001) applied evolutionary algorithm (EA) at the preprocessing stage to reduce the dimension/difficulty of the problem and to increase the learning efficiency in data mining. Hruschka and Ebecken (2000), and Meesad and Yen (2001) used EA at the post-processing stage to extract rules from a neural network.

Another promising approach to address the efficiency deficiency of EA in dealing with data mining problems is to exploit the inherent parallel nature of EA by formulating the problem into a distributed computing structure suitable for parallel processing, i.e., to divide a task into subtasks and to solve the subtasks simultaneously using multiple processors. This divide-and-conquer approach has been applied to EA in different ways and many parallel EA implementations have been reported in literature (Cantú-Paz, 1998; Goldberg, 1989, Rivera, 2001). Among the different parallel implementations of evolutionary algorithms, Levine (1995) developed the PGAPack, which is a parallel evolutionary algorithm library that supports global parallelization. The package is written in C and communication is carried out using MPI, which is a popular library specification for message passing. Andre and Koza, (1995) used the transputers hardware and Tomassini and Fernandez (2000) applied the MPI as a platform for implementing their parallel EAs. Tanev, *et al*., (2001) made use of Distributed Component Object Model (DCOM), which is a protocol that enables software components to communicate directly over a network. Meta Group Consulting, (1998) combines Java technology (Sun, 2001) with Common Object Request Broker Architecture (CORBA) to give good solutions in remote method invocation and remote class loader as a pure OOP language. Chong (1997) proposed an application of Java applet on the Internet which focuses on the massive distributed approach, and Distributed Resources Evolutionary Algorithm Machine (DREAM) is a project to provide the technology and software infrastructure necessary to support the next generation of evolving infohabitants in a way that makes the infrastructure universal, open and scalable (Paechter and Back, 2000). The availability of powerful-networked computers presents a wealth of computing resources that can provide the processing power required to solve those problems unsolvable in a single computer. Large problems can be divided into many smaller subtasks mapped into the individual computers available in the system. This potential computational power can be much stronger than a supercomputer. However, the heterogeneous hardware and software on the Internet, presents an insurmountable difficulties for the implementation of distributed systems especially in the area of portability, distribution and security. The recent emergence of Java technology, a fully object-oriented platform-neutral programming language by Sun Microsystems Inc., presents an opportunity for implementing such a distributed system efficiently (Sun, 2001).

1.4. Contributions

In this thesis, two rule-based classification algorithms are presented in which the first one is a two-phase evolutionary approach and the second is a distributed co-evolutionary classifier. The classification performances and the efficiency of the evolution process are the two major considerations of the both algorithms. In the two-phased approach, a hybrid evolutionary algorithm is utilized in the first phase to confine the search space by evolving a pool of good candidate rules, e.g., genetic programming is applied to evolve nominal attributes for free structured rules and genetic algorithm is used to optimize the numeric attributes for concise classification rules without the need of discretization. These candidate rules are then used in the second phase to optimize the order and number of rules in the evolution for forming accurate and comprehensible rule sets. Good simulation results on three medical datasets show that the algorithms can be used as an assistant tool in clinical practice for better understanding and prevention of unwanted medical events. While in the co-evolutionary system, by utilizing the existing Internet and hardware resources, distributed computing is naturally incorporated into the coevolutionary algorithm to enhance its concurrent processing and performance. Through the inter-communications between the different species (rules and rule sets), the cooperation is conducted in a more effective and efficient way. Rules thus generated are all crucial to the problem, which makes it easy to find the resultant rule set with a fairly good performance. The proposed distributed coevolutionary classifier is extensively validated upon 6 datasets obtained from UCI machine learning repository, which are representative artificial and real-world data from various domains. Comparison results show that the algorithm produces comprehensible and good classification rules for all the datasets, which are very competitive or better than many classifiers widely used in literature.

1.5. Thesis outline

This thesis consists of five chapters.

Chapter 2 describes the basic concept of rule induction and then deepen the idea by introducing how evolutionary computation can be resorted to doing rule induction.

Chapter 3 details the implementation of the two-phase evolutionary rule induction algorithm followed by the validation on three medical datasets.

The distributed coevolutionay system is presented in chapter 4, in which the design idea of the distributed system, the implementation detail of the algorithm and the evaluation results are all included.

Chapter 5 concludes the whole thesis and points out the direction of future research.

Chapter 2

Evolutionary computation in rule induction

2.1. Introduction to rule induction

Given a set of labeled instances, the objective of classification is to discover the hidden relations or regulations between attributes and classes. The classification rules are extracted in the hope that they can be used to automate classification of future instances. In the classification task, the discovered knowledge is usually represented in the form of decision trees or IF-THEN classification rules, which has the advantage of being a high-level and symbolic knowledge representation that contributes to the comprehensibility of the discovered knowledge. In this thesis, the knowledge is presented as multiple IF-THEN rules in a decision rule list or rule set. Such rules state that the presence of one or more items (antecedents) implies or predicts the presence of other items (consequences). A typical rule has the form of

$$\text{Rule: } \textit{IF } \boldsymbol{X_1} \textit{ and } \boldsymbol{X_2} \textit{ and } \ldots \boldsymbol{X_n} \textit{ THEN } \boldsymbol{Y},$$

where $\boldsymbol{X_i}$, $\forall\ i \in \{1, 2, \ldots, n\}$ is the antecedent that leads to a prediction of $\boldsymbol{Y}$, the consequence. Each of the IF-THEN rules can be viewed as an independent piece of knowledge. New rules can be added to an existing rule set without disturbing those already there, and multiple rules can be combined together to form a set of decision rules. The basic structure of the decision rule list could be built as follows,

```
IF antecedent_1 THEN class_1
ELSE IF antecedent_2 THEN class_2

…

ELSE class_default
```

When the rule list is evaluated or used to classify a new instance, the first rule (top most) will be considered first. If the rule does not match the instance (i.e., not able to classify the instance), the next rule will be considered. The matching process is repeated until a corresponding rule is found. In the case where none of the rules in the rule list matches the new instance, the new instance will be classified as the default class, which is usually the largest class in the data set.

The discovered decision rules can be evaluated according to several criteria, such as classification accuracy on unlabeled instances (testing set), degree of confidence in the prediction, comprehensibility and interestingness. Among these measures, classification accuracy is the major metric to evaluate the performance of a classifier. The comprehensibility measures how clear and easy a rule is for human to understand and take action on it accordingly. Generally, rules that are incomprehensible to human are often useless in data mining or knowledge discovery because such rules are not beneficial to the users.

2.2. Evolutionary computation in rule induction

Evolutionary algorithms for knowledge discovery in rule induction can be broadly divided into Michigan and Pittsburgh approaches (Michalewicz, 1994) depending on how rules are encoded in the population of individuals. In the Michigan approach, each individual encodes a single prediction rule. Examples of EAs for classification that follow the Michigan approach are REGAL (Giordana and Neri, 1995), GGP (Wong and Leung, 2000) and De Falco *et al*. (2002). In the Pittsburgh approach, each individual encodes a set of prediction rules (Freitas, 2002). Examples of EAs for classification that follow the Pittsburgh approach are GABIL (De Jong *et al*., 1993), GIL (Janikow, 1993) and BGP (Rouwhorst and Engelbrecht, 2000). The choice between the two coding approaches strongly depends on which kind of knowledge is targeted. For the task of classification, the quality of rule set is usually evaluated as a whole rather than the quality of a single rule, i.e., the interaction among the rules is important. In this case, the Pittsburgh approach is a better choice. On the other hand, the Michigan approach is more suitable for tasks where the goal is to find a small set of high-quality prediction rules, and each rule is evaluated independently of other rules (Noda *et al*., 1999).

The Pittsburgh approach directly takes into account rule interaction when computing the fitness function of an individual. However, the individuals encoded with this approach are syntactically longer, thus making the optimal solution difficult to be found when the search space is large. In Pittsburgh approach, standard genetic operators may need to be modified for coping with the relatively complex individuals to ensure feasibility of solutions. On the other hand, the Michigan approach encoded individuals are simpler and syntactically shorter, thus making the search of solutions easier and faster. However, this approach has some drawbacks, e.g., each rule is encoded and evaluated separately without taking into account interactions among different rules. Furthermore, the Michigan approach often needs to include niching methods such as token competition (Wong and Leung, 2000) in order to maintain the diversity of population or to converge to a set of rules instead of a single rule.

To utilize advantages of both approaches and to compromise the drawbacks of each, the two approaches can be applied together in a certain way. In fact, the final rule set can be obtained through a two-phase evolution process where, in the first phase, classification rules are learnt from the data sets via a Michigan coding approach, which produces a pool of candidate good rules for evolving the final rule set in the second phase. Since rule sets with different number of rules are targeted in the second phase, Pittsburgh coding approach is used to encode the individuals. In this approach, the optimal number of rules in a rule set can be decided automatically in the second phase. This is unlike many existing approaches such as Peña-Reyes and Sipper (1999; 2001) where the number of rules in a rule set often needs to be determined manually and *a-priori*. This two-phase evolutionary classifier also confines the usually large classification search space and consequently requires a smaller population and generation size. In this approach, the problem of the Pittsburgh's approach in finding the usually very large/infinite combination of classification rules can be reduced, i.e., the first phase searches for good rules in a complex search space and the second phase searches for optimal combination of rules obtained in the first phase. Good combination of rules are easier to be found in this way since the number of rules available is limited and the rules are all essential to the problem. For example, the population size and generation size was set as 200 and 2500, respectively in Peña-Reyes and Sipper (1999). In the two-phase approach, the population size is set as 100 in the first phase and 50 in the second phase, and the generation size is set as 100 and 50 in the first and second phase, respectively.

2.3. Coevolution

Coevolution refers to the simultaneous evolution of two or more species with coupled fitness (Liu *et al.*, 2001). Such coupled evolution favors the discovery of complex solutions (Paredis, 1995). Coevolution of species can either compete (e.g., to obtain exclusivity on a limited resource) or cooperate (e.g., to gain access to some hard-to-attain resource). In a competitive coevolutionary algorithm, the fitness of an individual is based on direct competition with individuals of other species, which in turn evolve separately in their own populations. Increased fitness of one of the species implies a diminution in the fitness of the other species. This evolutionary pressure tends to produce new strategies in the populations involved to maintain their chances of survival. This “arms race” ideally increases the capabilities of all species until they reach an optimum. For further details on competitive coevolution, readers may refer to Rosin and Belew, (1997).

Cooperative coevolutionary algorithms involve a number of independently evolving species, which together form a complex structure for solving difficult problems. The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favors the development of cooperative strategies and individuals. Single population evolutionary algorithms often perform poorly, manifesting stagnation, convergence to local optima and computational costliness, when they are confronted with problems presenting one or more of the following features: (1) the sought-after solution is complex, (2) the problem or its solution is clearly decomposable, (3) the genome encodes different types of values, (4) strong interdependencies among the components of the solution, and (5) components-ordering drastically affects fitness (Peña-Reyes and Sipper, 2001). Cooperative coevolution addresses these issues effectively, and consequently widening the range of applications in evolutionary computing. Paredis (1995) applied cooperative coevolution to problems that involved finding simultaneously the values of a solution and their adequate order. In his approach, a population of solutions coevolves alongside a population of permutations performed on the genotypes of the solutions. Moriarty (1997) used a cooperative coevolutionary approach to evolve neural networks. Each individual in one species corresponds to a single hidden neuron of a neural network and its connections with the input and output layers. This population coevolves alongside a second one whose individuals encode sets of hidden neurons (i.e., individuals from the first population) forming a neural network.

With coevolution on hand, the whole rule sets extraction process can be made to be an integrated one instead of using a two-phase approach stated in the previous section. By utilizing the coevolutionary algorithms, a rule population and several rule set populations can be evolved concurrently, which make the generated rules more relevant and useful for the rule set construction, so the rule sets with fairly good classification performance can be expected. Furthermore, since the different species (rules and rule sets) evolve in a parallel and cooperative way, the distributed technology can naturally be incorporated into the whole system to make the work done in a more efficient and effective manner.

2.4. Conclusion

In this chapter, an overall introduction of rule induction is given in the first section, from which one can learn what a rule set looks like and how to derive a rule set. Specifications of how evolutionary algorithms can be applied to rule induction are introduced in the following sections, which build a good foundation for the algorithms to be presented in the following chapters.

Chapter 3

A two-phase evolutionary rule induction algorithm

3.1 Algorithm overview

In this chapter, the classification task is formulated as a complex search optimization problem, where hidden relationships of the attributes to class are targeted knowledge to be discovered. The candidate solution that is in the form of a comprehensible Boolean rule set is obtained through a two-phase evolution mechanism as shown in Figure 3.1. The first phase searches for a pool of good candidate rules using Michigan coding approach (Michalewicz, 1994), while the second phase finds the best Boolean rule set by evolving and forming rule sets from the pool of rules. Since rule sets with different number of rules are targeted in the second phase, Pittsburgh coding approach (Michalewicz, 1994) is used to encode the individuals. In the proposed evolutionary classifier, the optimal number of rules in a rule set is decided automatically in the second phase, which is advantageous to many approaches where the number of rules in a rule set often needs to be determined a priori (Joshi *et al*, 2001, Peña-Reyes and Sniper, 1999; 2001).

The proposed two-phase evolutionary classifier also confines the usually large generation size. In this way, the inherent problem of Pittsburgh's coding method in finding the usually large combination of classification rules is greatly reduced, e.g., it is relatively easy to find good combination of rules in the second phase since the number of rules obtained in the first phase is confined and essential to the problem. For example, the population and generation size was set as 200 and 2500, respectively in (Peña-Reyes and Sniper, 1999). In this approach, the population size is set as 100 in the first phase and 50 in the second phase, and the generation size is set as 100 and 50 in the first and second phase, respectively.

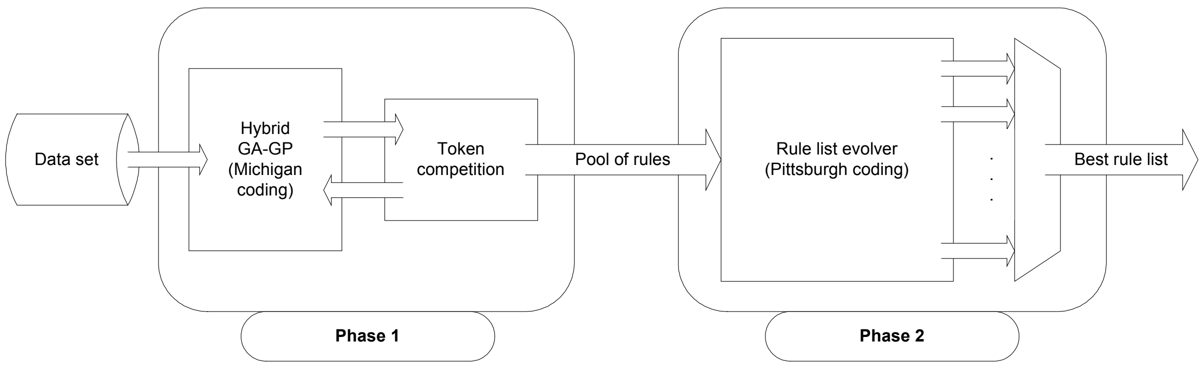

Data set
Hybrid GA-GP (Michigan coding)
Token competition
Pool of rules
Rule list evolver (Pittsburgh coding)
Best rule list
Phase 1
Phase 2

Figure 3.1: Overview of the two-phase hybrid evolutionary classifier

A great challenge of applying evolutionary algorithms in data mining problems is that sometimes an algorithm should have the capacity of dealing the numeric and nominal attributes simultaneously. To be equipped with this capability, Bojarczuk *et al*., (2000) implemented a non-standard tree structure GP. In their algorithm, functions are constructed via Boolean operators, and terminal sets are chosen based on booleanized attributes. The numeric attributes in this approach are discretized into nominal boundaries *a-priori* in order to use the booleanized attributes. Nevertheless, this greatly restricts the search capability of GP, i.e., the classification outcome depends on how well the boundaries were defined.

One possible approach of handling both nominal and numeric attributes in data classification is through the hybridization of GA and GP. Howard and D'Angelo (1995) proposed a hybrid GA and GP called Genetic Algorithm-Program (GA-P) which has been applied to evolve expressions for symbolic regression problem. In their approach, GP is used to construct expression tree while GA is used to construct numeric constant and coefficient of nominal attributes used in the expression. Unlike GA-P that was designed to solve regression problems, an effective approach of fusing GA and GP for different targeted application of mining comprehensible classification rules is proposed in this chapter. Although both approaches utilize the concept of GA and GP hybridization, the GA-P is designed for predictive application, which is different from the problem of data classification addressed in this thesis. Moreover, the GA-P treated the fixed binary string of GA and tree expression of GP as 2 inseparable entities in a chromosome. In this approach, the string in GA and tree structure in GP is indirectly related and each can work without another, e.g., if there is only numeric attributes in the targeted problem, only the GA part will be fired in our approach. In addition, a two-phase evolutionary process is adopted in this approach, i.e., the hybrid evolutionary algorithm is applied to generate good rules in the first phase, which are then used to evolve comprehensive rule sets in the second phase.

3.2 Phase 1: The Hybrid GA-GP

The evolutionary classifier, namely EvoC, has been implemented and integrated into the Java-based public domain data mining package 'WEKA' (Garner, 1995; Witten and Frank, 1999). Figure 3.2 shows the phase one of the program flowchart of EvoC, where the initial population is created from the training set. The attributes in the training set are built into nominal and numeric table for GP and GA, respectively. Each individual encodes a single rule and the population is structured such that all individuals are associated to the same class. This structure avoids the need of encoding the class values, i.e., the THEN part of the rules is encoded implicitly in the individuals.

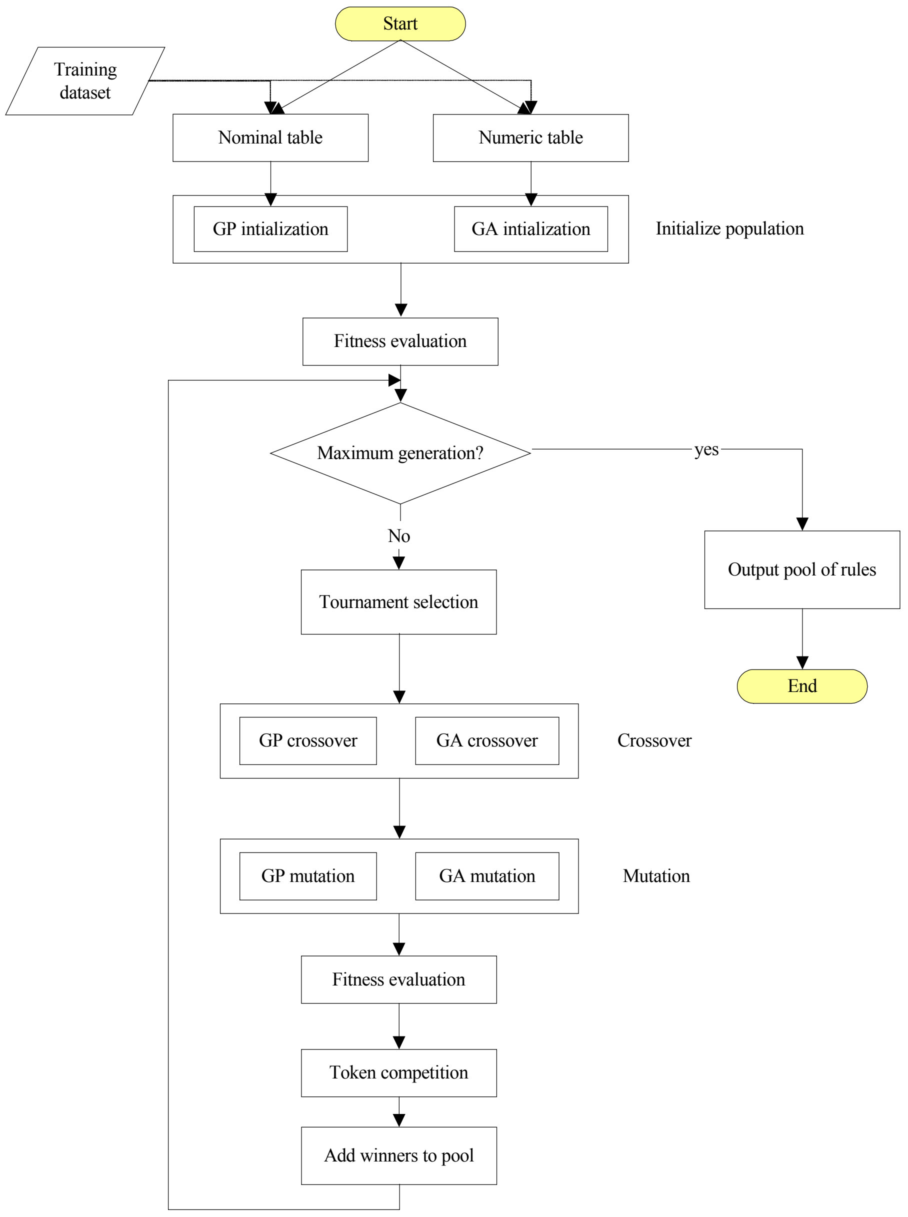


Figure 3.2: The program flowchart for the phase one of EvoC

The tournament selection scheme (Banzhaf, 1998) with a tournament size of 2 is implemented in EvoC. When a new population is formed, the token competition (Wong and Leung, 2000) is applied as a covering algorithm to penalize redundant individuals as well as to retain individuals that cover the problem space well (Hu, 1998). The winners in the token competition will be added to a pool of candidate rules and the pool is maintained such that no redundant rules may exist and the previously encountered good rules are kept for subsequent competitions. All individuals in the pool including the current population are participated in the token competition in order to ensure that no redundant rules exist in the pool.

The evolutionary process in phase one is run for every class of data set. For an *n* class problem, there will be *n* evolutionary iterations. The pool of rules for every class is combined into a global pool, which will be presented as the input to the second phase. Each individual in the first phase contains two different chromosome structures, which are treated separately in the evolution and assigned to handle the nominal and numeric attributes. The chromosome structure, genetic operations and handling techniques of EvoC in phase one are described in the following subsections.

3.2.1 Chromosome Structure and Genetic Operations

The weather data set shown in Table 3.1 is used as an example to show how the proposed hybrid GA-GP works on different types of attributes. The objective of the data set is to learn whether a specific game can be played on a given weather. Each of the columns in Table 3.1 represents an attribute. The last column is the class attribute to be learned. Each of the rows represents an instance, and the collection of instances forms the data set. For the weather data set, the “Outlook” and “Windy” are nominal attributes, while the “Temperature” and “Humidity” are numeric attributes.

The genetic programming tree-based chromosome representation has been used to encode nominal attributes that are Booleanized, and many logical operators have been applied to evolve highly flexible solutions in classification problems (Bojarczuk *et al*, 2000; Tan *et al*, 2002a). However, the difficulty of integrating general arithmetic operators with Booleanized attributes in GP limits the flexibility of handling real-world data that often consists of both the nominal and numeric attributes (Hu, 1998). One passive approach is to discretize the numeric attributes into boundaries at the expense of lower classification accuracy for the rules found. To address this problem, the approach of having independent chromosomes to handle numeric attributes in data classification is adopted in this paper. Since the representation of fixed-length chromosome with numeric genes in genetic algorithms is well suited for numerical optimization (Goldberg, 1989) it is used in EvoC to deal with the numeric attributes in the classification.

Table 3.1 The weather data set

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
Sunny	85	85	FALSE	No
Sunny	80	90	TRUE	No
Overcast	83	86	FALSE	Yes
Rainy	70	96	FALSE	Yes
Rainy	68	80	FALSE	Yes
Rainy	65	70	TRUE	No
Overcast	64	65	TRUE	Yes
Sunny	72	95	FALSE	No
Sunny	69	70	FALSE	Yes
Rainy	75	80	FALSE	Yes
Sunny	75	70	TRUE	Yes
Overcast	72	90	TRUE	Yes
Overcast	81	75	FALSE	Yes
Rainy	71	91	TRUE	No

A. GP chromosome structure: The selection of functions and terminals is the preparatory step in genetic programming (Koza, 1992). In EvoC, two Boolean operators are adopted as functions, i.e., ‘AND’ and ‘NOT’. These two functions are sufficient to build a basic classification rule in the form of “*IF* antecedent$_1$ AND (NOT antecedent$_2$) AND … *THEN* consequence”. The classification rules that are built of ‘AND’ and ‘NOT’ can then be combined to form the decision rule set (the ‘OR’ effect). The terminal set contains all possible *attribute-value* pairs for a given data set. For example, the possible attribute-value pairs could be outlook-overcast, outlook-sunny, outlook-rainy, windy-TRUE or windy-FALSE for the weather data set. To avoid redundant or conflicting nodes exist in the same tree, these terminals are built into a table and only one *attribute-value* pair can be selected from each attribute entry for a tree structure. The initial population in GP is created with the approach of ‘*ramped-half-and-half*’ (Koza, 1992).

B. GA chromosome structure: The fixed-length real-coding chromosome structure is adopted in GA (Goldberg, 1989). The range of each numeric attribute is represented by two real-coded genes: one encodes the upper bound and the other the lower bound. As depicted in Figure 3.3, the M^{th} ($M \leq N$) and $(M + N)^{\text{th}}$ genes encode the range of the M^{th} numeric attribute. In the initial population, the lower and upper bound of each attribute is initialized as the corresponding minimum and maximum, respectively. For example, in the weather problem, the minimum and maximum of the two numeric attributes “Temperature” and “Humidity” is (64, 85) and (70, 96), respectively. Therefore the initialization of chromosomes is given as (64, 70, 85, 96). Obviously, such an approach starts the evolution with generality and subsequently searches for specificity. Since nominal attributes consist of a finite number of values, their hidden relationships are often easier to be discovered. Based on this assumption, the above initialization is adopted to give nominal attributes a higher priority. As the evolution process, the range of certain numeric attributes will shrink and the corresponding nominal parts will improve accordingly to produce better classification accuracy.

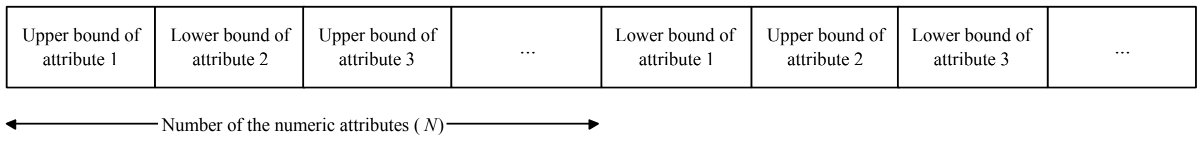

Upper bound of attribute 1
Lower bound of attribute 2
Upper bound of attribute 3
...
Lower bound of attribute 1
Upper bound of attribute 2
Lower bound of attribute 3
...
Number of the numeric attributes (N)

Figure 3.3: The chromosome structure in GA

C. Mutation and crossover: Since the GP in EvoC only deals with nominal attributes, standard tree-based crossover and mutation operators are employed in the GP (Koza, 1992). However, a specialized mutation operator is used in GA in order to avoid annoyance rules such as age ≤ 45.23. The values of every numeric attributes of the data set to be learned are stored in a table, and the mutation is performed by fetching a random corresponding value from the table and replaces the value of the attribute in the chromosome. Standard single-point crossover where two parents exchange their genes from a random position to reproduce the offspring is adopted in GA (Goldberg, 1989). It should be noted that these genetic operators utilized data in the available data sets for the lower and upper bounds of the attributes, which not only makes the best use of information in the data sets, but also guarantees the meaningfulness of the final rules produced by EvoC.

3.2.2 Automatic Attribute Selection

Although a data set often contains many attributes, it is common that only a fraction of the attributes will appear in a single rule. For example, a rule for the weather data set may be in the form of "*IF outlook = sunny and humidity ≥ 83, THEN play = no*", where only two out of the five attributes are considered in this rule. This characteristic of rules seems to be counterintuitive to the fixed-length chromosome structure of GA, where all numeric attributes are considered in the evolution. If chromosomes in the GA are converted to rules directly, all the numeric attributes will be included in the rules, which may result in the redundant rules. Such contradiction, however, could be overcome by studying the characteristic of chromosomes in GA for succinct presentation of rules.

Suppose a candidate individual in the solution produces a rule in the form of "*IF outlook = sunny and* $64 \le temperature \le 85$ *and* $83 \le humidity \le 96$, *THEN play = no*". Since '64' and '85' are the lower and upper limits of the temperature, it casts no restrictions on all data samples and thus the temperature condition will be discarded from the rule. This observation is also applicable to the humidity attribute. Since '96' is the upper limit of humidity, all the instances whose humidity are higher than '83' will satisfy this condition. Hence, '96' is unnecessary and will also be excluded from the rule. After these operations, the final concise rule becomes "*IF outlook = sunny and humidity ≥ 83, THEN play = no*".

3.2.3 Fitness Function

When a rule or individual is used to classify a given training instance, one of the four possible concepts can be observed: true positive (*tp*), false positive (*fp*), true negative (*tn*) and false negative (*fn*). The true positive and true negative are correct classifications, while false positive and false negative are incorrect classifications. For a 2-class case, with class 'yes' and 'no', the four concepts can be easily understood with the following descriptions,

- True positive: the rule predicts that the class is 'yes' (positive) and the class of the given instance is indeed 'yes' (true);
- False positive: the rule predicts that the class is 'yes' (positive) but the class of the given instance is in fact 'no' (false);
- True negative: the rule predicts that the class is 'no' (negative) and the class of the given instance is indeed 'no' (false);
- False negative: the rule predicts that the class is 'no' (negative) but the class of the given instance is in fact 'yes' (true).

Using these concepts, the fitness function used in the first phase of EvoC is defined as,

$$fitness = w \times \frac{tp}{(tp + fn)} \times \left(1 + \frac{tn}{(tn + fp)}\right) \tag{1}$$

$$\text{with} \quad w = \frac{N}{N + fp} \tag{2}$$

where N is the total number of instances in the training set and w is a penalty factor. The value of the fitness function is in the range of 0 to 2. The fitness value is 2 (the fittest) when all instances are correctly classified by the rule, i.e., when fp and fn are 0. A penalty factor w that tends to minimize fp is included in the fitness function to evaluate the quality of the combined individuals in the rule set. This is because Boolean sequential rule list (where rules are considered one after another) is very sensitive and tends to have a large number of false positives (*fp*) due to the virtual 'OR' connections among the rules, e.g., when a rule with large *fp* is considered first in a rule list, many of the instances will be classified incorrectly.

3.2.4 The Covering Algorithm

The covering algorithm employs the token competition (Wong and Leung, 2000) to promote the diversity and to evolve multiple rules in the first phase of EvoC. Multiple rules that cover the same instances in the training set often increase the tendency of premature convergence in the evolution. In most cases, only a few of these multiple rules are useful and cover most of the instances while others are redundant. To achieve the optimal performance for the rule list evolver in the second phase, all rules that are able to cover at least one instance in phase one will be retained in the pool of candidate rules, which are maintained by the covering algorithm.

3.3 Phase 2: The Rule Set Evolver

In the phase one of EvoC, the hybrid GA-GP approach is applied to find good classification rules in a usually complex search space. The approach is a Michigan-style algorithm where classification performance of the rule set is not needed for fitness evaluations. Although the token competition can serve as a rule selection mechanism, e.g., rules that fail to seize any token (a token represents an example in the dataset) will be eliminated, a Pittsburgh-like approach is required in the second phase in order to find the optimal order and number of rules in a rule set from the pool of candidate rules evolved in phase one. To determine the optimal number of rules in a rule set, the population in phase two is divided into several sub-populations, where each sub-population is dedicated to optimize the order of rules with a given rule number. For example, if the number of rules in the candidate pool is *n*, then there will be *n* sub-populations and the i^{th} sub-population will be evolved to optimize the rule set containing *i* rules.

After the initialization, each sub-population will be evolved independently and there is no interaction among the sub-populations. At the end of the evolution, each sub-population outputs its 'best' candidate rule set, which will compete (based on the classification accuracy) with the 'best' rule sets generated by other sub-populations to obtain the final optimal rule set. In this approach, the order and number of rules in the rule sets can be optimized and determined simultaneously. To retain concise rule sets in the classification, a shorter rule set is preferable to a longer one even if both achieved the same classification accuracy. All rules obtained in the first phase of EvoC are given an index and these rules will be selected randomly to build up the rule sets. Figure 3.4 depicts the initialization of two chromosomes having 3-rules set and 6-rules set respectively. Similar to the GA in phase one of EvoC, standard single-point crossover and tournament selection schemes are adopted in phase two. The mutation operation is performed by randomly selecting a rule from the pool of candidate rules to replace the rule for mutation. In phase two, the fitness function considers the classification accuracy on the training set as given by,

$$fitness = \frac{tp + tn}{N} \tag{3}$$

where N is the total number of instances in the training set.

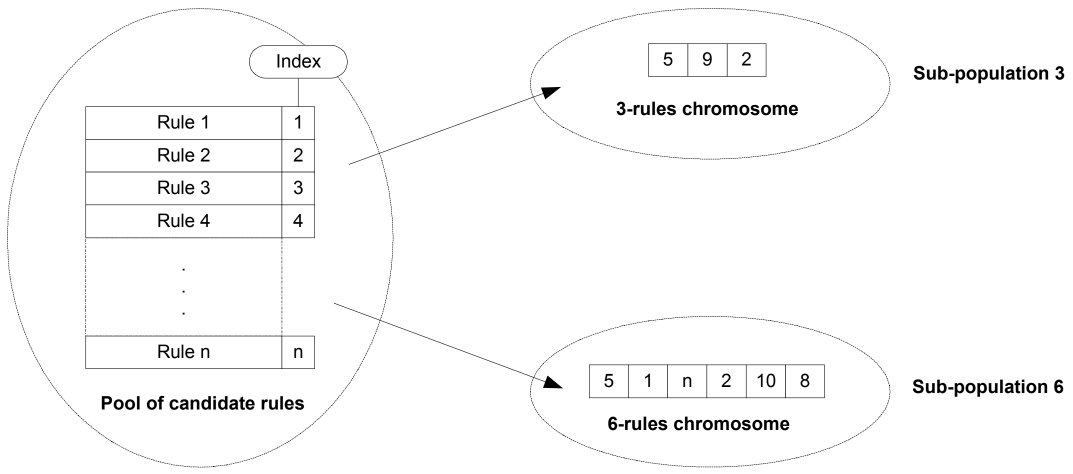


Figure 3.4: Example of chromosomes initialization in the phase two of EvoC

3.4 Applications on medical diagnosis

Clinical medicine is facing a challenge of knowledge discovery from the growing volume of data. Nowadays enormous amounts of information are collected continuously by monitoring physiological parameters of patients. The growing amounts of data has made manual analysis by medical experts a tedious task and sometimes impossible. Many hidden and potentially useful relationships may not be recognized by the doctors or physicians. The explosive growth of data requires an automated way to extract useful knowledge. One of the possible approaches to this problem is by means of data mining or knowledge discovery from database (KDD) (Brameier and Banzhaf, 2001). Through data mining, interesting knowledge and regularities can be extracted and the discovered knowledge can be applied in the corresponding field to increase the working efficiency and to improve the quality of decision making.

Classification rules are typically useful for medical problems and have been massively applied particularly in medical diagnosis. Such rules can be verified by the experts and may provide better understanding of the problem in-hand. Numerous techniques have been applied to data mining applications over the past few decades, such as expert systems, artificial neural networks, linear programming, database systems, and evolutionary algorithms (Chang *et al*., 1999; Kupinski and Anastasio, 1999; Setiono, 1996; Witten and Frank, 1999; Wong *et al*., 2000). Among these approaches, evolutionary algorithms (EAs) have been emerged as a promising technique in dealing with the increasing challenge and problems in medical domain. Recently, EAs has been utilized at different stages of knowledge discovery process in medical data mining applications. Kim and Han (2000), and Liu *et al*., (2001) applied genetic algorithm at the preprocessing stage to reduce the dimension/difficulty of the problem and to increase the learning efficiency in data mining. Hruschka and Ebecken (2000), and Meesad and Yen (2001) used genetic algorithm at the post-processing stage to extract rules from a neural network. Other EAs approaches for generating classification rules in data mining include Wang *et al*., (1998), Congdon, (2000), and Fidelis *et al*., (2000). In the following sections, the proposed EvoC is applied to three real world medical datasets, which justifies EvoC as a useful tool to aid the prognosis and diagnosis of diseases.

3.4.1 The Medical Diagnosis Data Sets

The medical diagnosis data sets used in this study are the hepatitis data set and breast-cancer diagnosis databases obtained from University of California, Irvine (UCI) machine-learning repository at http://www.ics.uci.edu/~mlearn/MLRepository.html. The Hepatitis data set was collected at Carnegie-Mellon University (Cestnik, *et al*, 1987) and donated to UCI ML repository in 1988. The two breast-cancer diagnosis data sets, i.e., *Wisconsin Breast Cancer Database* (WBCD) and *Wisconsin Diagnostic Breast Cancer* (WDBC), were collected at different periods of time with different attributes recorded (Street *et al*, 1993). The former was donated to UCI ML repository in 1991, while the latter was in 1995 for public access. For both breast cancer data sets, the classification task is to determine case of benign or malignant from the physical attributes of cell given in the data sets. The characteristics of these data sets are briefly described as follows:

A. The Hepatitis Data Set (HEPA): The Hepatitis data set is summarized in Table 3.2, which consists of 155 instances. Each instance consists of 19 attributes, namely age, sex, steroid, antivirals, fatigue, malaise, anorexia, liver big, liver firm, spleen palpable, spiders, ascites, varices, bilirubin, alk phosphate, SGOT, albumin, protime and histology. This problem includes both nominal and numeric attributes, which is particularly suitable for verifying the performance of EvoC. The HEPA is a complex and noisy data set since it contains a large number of missing data. The class is distributed with 32 (20.65%) DIE samples and 123 (79.35%) LIVE samples. The classification task is to predict whether a patient with hepatitis will live or die.

Table 3.2 Summary of the HEPA data set

Attribute	Possible values
Age	Integer 1 – 80
Sex	Male, Female
Steroid	No, Yes
Antivirals	No, Yes
Fatigue	No, Yes
Malaise	No, Yes
Anorexia	No, Yes
Liver BIG	No, Yes
Liver firm	No, Yes
Spleen palpable	No, Yes
Spiders	No, Yes
Ascites	No, Yes
Varices	No, Yes
Bilirubin	0.39, 0.80, 1.20, 2.00, 3.00, 4.00
Alk phosphate	33, 80, 120, 160, 200, 250
SGOT	13, 100, 200, 300, 400, 500
Albumin	2.1, 3.0, 3.8, 4.5, 5.0, 6.0
Protime	10, 20, 30, 40, 50, 60, 70, 80, 90
Histology	No, Yes
Class	Die (20.65%), Live (79.35%)

B. The Wisconsin Diagnostic Breast Cancer (WDBC): The WDBC data set is summarized in Table 3.3 and consists of 569 instances. Each instance consists of 10 real-valued attributes of the nuclear for the cancer cell, namely radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry and fractal dimension. These attributes are modeled such that higher values are typically associated with malignancy. The mean, worst (mean of the three largest values), and standard error of each attribute were computed for the original data set, resulting in a total of thirty attributes. In this study, however, only the mean values were considered in the rule extraction process. Detailed description of these 10 attributes is available from (Street *et al*, 1993). All the instances have been properly recorded and there is no missing value in this data set. The diagnosis class is distributed with 357 (62.7%) benign samples and 212 (37.3%) malignant samples.

Table 3.3 Summary of the WDBC data set

Attribute	Possible values	Description
Radius	Real	Mean of distances from center to points on perimeter
Texture	Real	Standard deviation of gray-scale values
Perimeter	Real	-
Area	Real	-
Smoothness	Real	Local variation in radius lengths
Compactness	Real	Perimeter^2 / area - 1.0
Concavity	Real	Severity of concave portions of the contour
Concave points	Real	Number of concave portions of the contour
Symmetry	Real	-
Fractal dimension	Real	"Coastline approximation" – 1
Diagnosis	Benign (62.7%), Malignant (37.3%)	

C. The Wisconsin Breast Cancer Database (WBCD): The WBCD data set is summarized in Table 3.4 and consists of 699 instances taken from fine needle aspirates (FNA) of human breast tissue. Each instance consists of nine measurements (without considering the sample's code number), namely clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. The measurements are assigned an integer value between 1 and 10, with 1 being the closest to benign and 10 the most anaplastic. Associated with each sample is its class label, which is either benign or malignant. This data set contains 16 instances with missing attributes' values. Since many classification algorithms have discarded these data samples, for the ease of comparison, the same way is followed and the remaining 683 samples are taken for use. Therefore the class is distributed with 444 (65.0%) benign samples and 239 (35.0%) malignant samples.

Table 3.4 Summary of the WBCD data set

Attribute	**Possible values**
Clump thickness	Integer 1 – 10
Uniformity of cell size	Integer 1 – 10
Uniformity of cell shape	Integer 1 – 10
Marginal adhesion	Integer 1 – 10
Single epithelial cell size	Integer 1 – 10
Bare nuclei	Integer 1 – 10
Bland chromatin	Integer 1 – 10
Normal nucleoli	Integer 1 – 10
Mitoses	Integer 1 – 10
Class	Benign (65.5%), Malignant (34.5%)

3.4.2 Simulation Settings

The EvoC was implemented in Java programming based on the Java Developers Kit (JDK 1.3.1) from Sun Microsystems. The simulations were performed using an Intel Pentium III 933 MHz processor with 512 MB SDRAM. To ensure the validity and replicability of the results, all experiments were designed carefully and all data sets used by the EvoC were partitioned into two sets: a training set and a testing set (or validation set). As indicated by Prechelt (1995), the fuzzy specification of the partitioning of training versus testing data is a big obstacle to reproduce or compare published machine-learning results. It is insufficient to only indicate the number of examples for each set in the partition since the experimental results may vary significantly for different partitions even if the numbers in each set are the same (Yao and Liu, 1997). In this work, a total of 100 simulation runs were performed for each of the three medical data sets, and a random seed[1] that is similar to the number of runs (i.e., the 50th simulation run uses a random seed of 50) was used to

[1] The random number generator used in the experiments is provided by Sun's JDK 1.3.1 and the data set randomizer used is provided by WEKA. Different partitioning of data sets might be resulted under different programming environments.

randomize the orders of data in the data sets. Each randomized data set was then partitioned into 66% of training data and 34% of testing data as follows:

- For the hepatitis data set, the first 102 examples are used for the training set and the remaining 53 examples for the testing set;
- For the WDBC data set, the first 376 examples are used for the training set and the remaining 193 examples for the testing set;
- For the WBCD data set, the first 451 examples are used for the training set and the remaining 232 examples for the testing set.

Table 3.5 lists the parameter settings of EvoC used in the simulations. The maximum initial depth and maximum crossover depth are GP specified control parameters, which are used to control the complexity of GP trees during the evolution. The parameter settings in Table 3.5 were applied to all experiments in this work, which should not be taken as the optimal set of parameters for each problem, but rather a generalized one for which the EvoC performs well over a number of different data sets.

Table 3.5 The setting of parameters in EvoC

Parameters	**Parameter description**	**Phase 1**	**Phase 2**
MaxInitDepth	The permitted depth of GP tree in initialization	6	-
MaxCrossoverDepth	The permitted depth of GP tree after crossover	17	-
ReproductionProb	The probability of an individual that will be copied to the next generation without changes	0.1	-
MutationProb	The probability of mutation	0.5	0.1
CrossoverProb	The probability of crossover	0.9	0.8
MaxGeneration	The generation number for the evolution	100	50
PopulationSize	The population size for the evolution	100	50

3.4.3 Simulation Results

Table 3.6 summarizes the classification results produced by EvoC over the 100 independent simulation runs for both the training and testing data sets. To obtain a better understanding of the classification performances for the different simulations, the histograms that summarize the experiment results of the three data sets are shown in Figure 3.5-3.7. For all the histograms, the classification performance axis indicates the classification accuracy achieved by the different number of rule sets obtained over the 100 independent simulation runs.

Table 3.6 Summary of the results in EvoC over the 100 independent simulation runs

Classification accuracy	**HEPA**	**WDBC**	**WBCD**
Training			
Max	90.20%	96.28%	99.33%
Min	79.41%	91.22%	96.23%
Mean	85.04%	94.36%	97.80%
StdDev	1.76%	0.91%	0.51%
Testing			
Max	94.34%	96.37%	99.13%
Min	75.47%	88.60%	95.26%
Mean	83.92%	93.04%	97.57%
StdDev	4.03%	1.47%	0.85%
Avg # rules	2.93	9.74	5.99

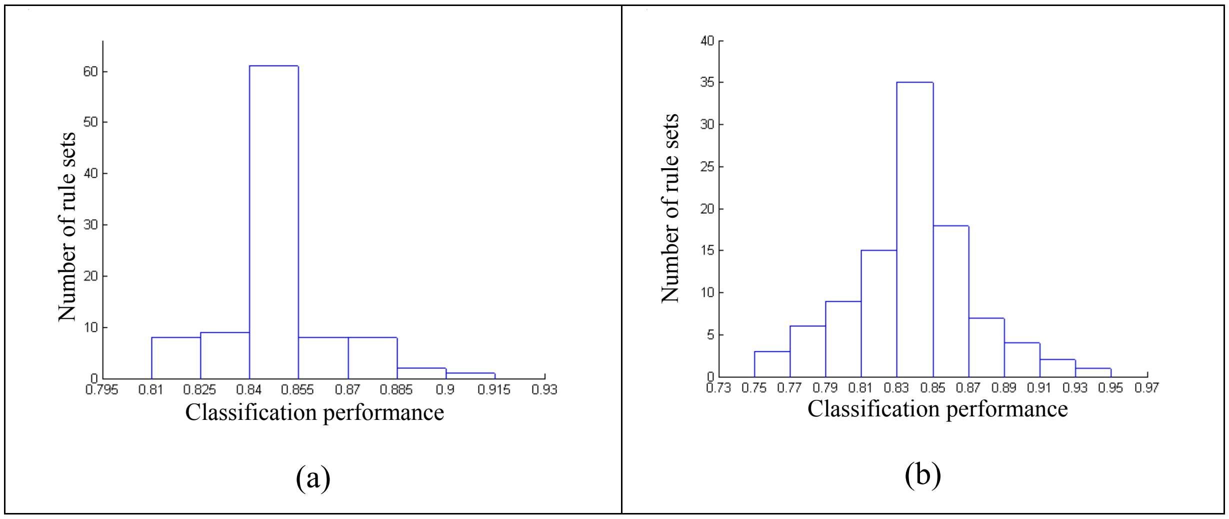


Figure 3.5: The performance of EvoC for the HEPA problem (a) training (b) testing

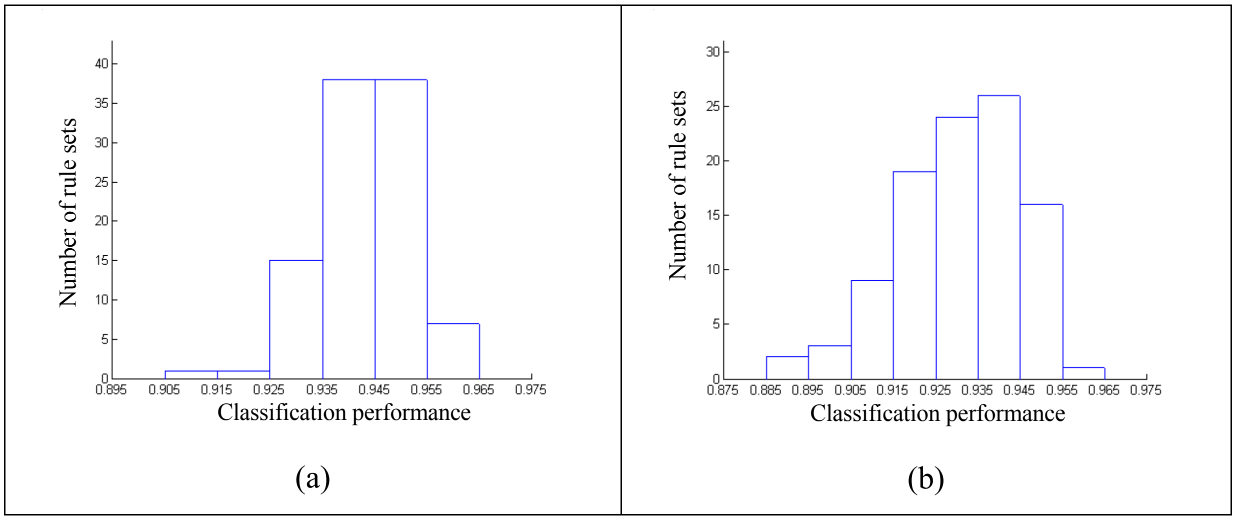


Figure 3.6: The performance of EvoC for the WDBC problem (a) training (b) testing

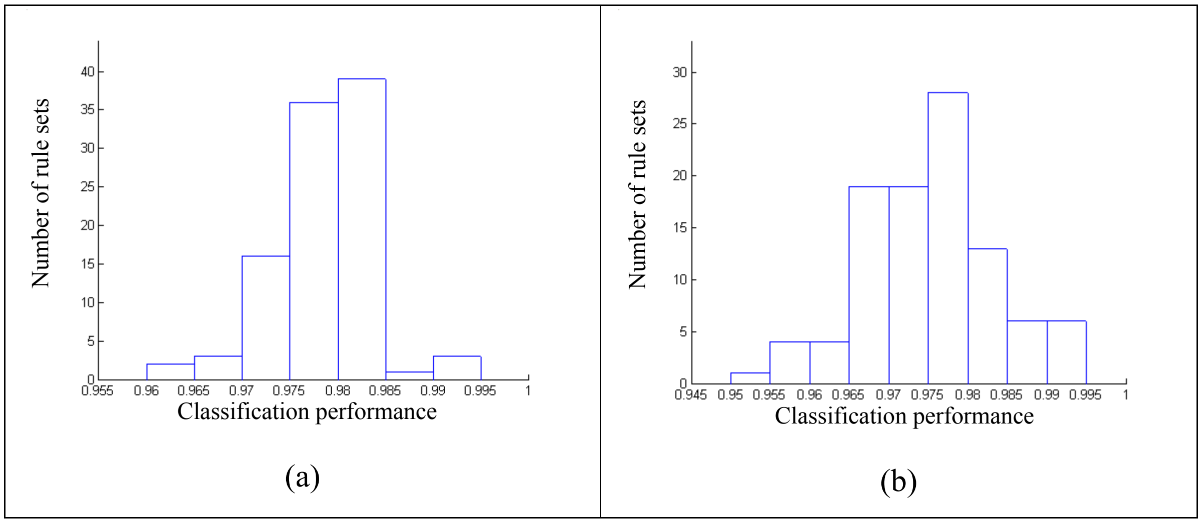


Figure 3.7: The performance of EvoC for the WBCD problem (a) training (b) testing

Tables 3.7-3.9 list the classification rules having the highest predictive accuracy (i.e., the classification accuracy on the testing data set) for the three medical data sets. Besides the fitness value, support factor and confidence factor are also provided to measure the performance of each rule. The support factor measures the coverage of a rule, which is the ratio of the number of instances covered by the rule to the total number of instances. The confidence factor measures the accuracy of a rule. For a rule "*IF X THEN Y*" and a training set of *N* instances, the support factor and confidence factor are given as,

$$\text{support} = \frac{\text{number of instances with both } X \text{ and } Y}{N} \tag{4}$$

$$\text{confidence} = \frac{\text{number of instances with both } X \text{ and } Y}{\text{number of instances with } X} \tag{5}$$

A careful examination of the relationship between the predictive accuracy of a rule set and its number of rules reveals an interesting finding. The rule sets with a large number of rules will not necessarily lead to high predictive accuracy, although they generally provide good performances on the training sets. It can also be observed that the first few rules in a rule set often cover a large portion of the samples and left relatively few samples for the remaining rules. Therefore when the data set is not noise-free, a large number of rules may cause over-fitting and leads to poor generalization. For example, in the WBCD problem, all of the best 6 rule sets that achieve a predictive accuracy of above 99% only contain an average of 4 rules. However, the 4 largest rule sets (all of which contain more than 15 rules) only produce an average accuracy of 97.75% on the testing samples.

Table 3.7 The best rule set of HEPA with an accuracy of 94.34%

	Rule	**Fitness**	**Support factor**	**Confidence factor**
1	IF FATIGUE = yes AND AGE >= 30.0 AND ALK_PHOSPHATE <= 280.0 AND ALBUMIN <= 4.3 AND PROTIME <= 46.0 THEN **Class** = DIE	1.2338	0.1961	0.5128

2	IF ANOREXIA = no
	AND BILIRUBIN <= 1.8
	AND SGOT <= 420.0
	THEN **Class** = LIVE
		1.0912	0.5588	0.8636

3	IF SPIDERS = yes AND AGE >= 30.0 AND 62.0 <= ALK_PHOSPHATE <= 175.0 AND ALBUMIN <=4.3 AND PROTIME <=85.0 THEN **Class** = DIE	1.2989	0.1765	0.6667
4	ELSE **Class** = LIVE			

Table 3.8 The best rule set of WDBC with an accuracy of 96.37%

	Rule	Fitness	Support factor	Confidence factor
1	IF Radius <= 14.95 AND Perimeter <= 116.1 AND Concavity <= 0.313 AND Concave_points <= 0.04908 THEN Diagnosis = benign	1.6774	0.5479	0.9763
2	IF Radius >= 13.0 AND Texture >= 15.76 AND Perimeter >= 74.72 AND Area >= 572.6 AND Concavity >= 0.03885 AND Concave_points >= 0.02402 THEN Diagnosis = malignant	1.5315	0.3032	0.8769
	IF Radius <= 17.01 AND Perimeter <= 116.1 AND Concavity <= 0.1122 AND Concave_points <= 0.1265 THEN Diagnosis = benign	1.5876	0.5931	0.8956
3	ELSE Diagnosis = malignant			

Table 3.9 The best rule set of WBCD with an accuracy of 99.13%

	Rule	Fitness	Support factor	Confidence factor
1	IF Clump_Thickness <= 8.0 AND Cell_Shape_Uniformity <= 8.0 AND Marginal_Adhesion <= 3.0 AND Bare_Nuclei <= 5.0 AND Bland_Chromatin <= 7.0 AND Normal_Nucleoli <= 8.0 THEN **Class** = benign	1.8702	0.6186	0.9789

2	IF Cell_Shape_Uniformity >= 3.0
	AND Single_Epi_Cell_Size >= 2.0
	AND Bland_Chromatin >= 2.0
	THEN **Class** = malignant
		1.739	0.3459	0.8571

3	IF Clump_Thickness <= 8.0 AND Cell_Size_Uniformity <= 4.0 AND Bland_Chromatin <= 3.0 AND Normal_Nucleoli <= 9.0 AND Mitoses<=1.0 THEN **Class** = benign	1.783	0.5898	0.9779
6	ELSE **Class** = malignant			

3.4.3 Performance Comparisons

This section compares the performance of EvoC with three popular machine-learning algorithms, i.e., *C4.5*, *PART* and *Naïve Bayes*. . The first two algorithms are chosen due to their rule-based characteristics as offered in EvoC. Comparisons between these two algorithms and EvoC include the performance of classification accuracy and rule set size (i.e., the number of rules in a rule set), since a good rule set should be both accurate and succinct. The method of Naïve Bayes is included here since it is a well-known statistical classifier that often gives high classification accuracy and provides good comparison to EvoC in terms of classification ability. Besides the comparisons of average results and standard deviations over the 100 simulation runs, a paired t-test (Montgomery et al, 2001) has also been performed between EvoC and the three algorithms respectively. The P-values are computed for testing the null hypothesis that the means of the paired observations on the accuracy rate are equal. These algorithms are briefly described below,

- The C4.5 proposed by Quinlan (1993) is a landmark decision tree program that has been widely used in practice;
- The PART is a rule-learning scheme capable of generating classification rules (Frank and Witten, 1998);
- The Naïve Bayes utilizes the Bayesian techniques, which has been studied by many machine-learning researchers (John and Langley, 1995).

In addition, the best results for the three data sets available in the literature[2] according to the author's best knowledge are also provided in the comparisons. Table 3.10 lists the P-values of the paired t-tests against the algorithms of C4.5, PART and Naïve Bayes for the three data sets. As can be seen, the P-values are rather small showing that the EvoC has outperformed the approaches of C4.5, PART and Naïve Bayes with a great confidence.

Table 3.10 The P-values of the paired t-tests against C4.5, PART and Naïve Bayes

	HEPA	**WDBC**	**WBCD**
C4.5 (Quinlan, 1993)	1.362×10^{-12}	3.545×10^{-02}	1.607×10^{-32}
PART (Frank and Witten, 1998)	5.565×10^{-10}	2.364×10^{-03}	2.990×10^{-30}
NaïveBayes (John and Langley, 1995)	0.313	2.644×10^{-09}	1.302×10^{-17}

A. Comparison Results for the HEPA Data Set

Wang *et al.* (2000) proposed an evolutionary rule-learning algorithm, called GA-based Fuzzy Knowledge Integration Framework (GA-based FKIF), which utilized genetic algorithms to generate an optimal or near-optimal set of fuzzy rules and membership functions from the initial population of knowledge. As shown in Table 3.11, only the best result produced by this algorithm is compared with EvoC since the average performance of GA-based FKIF was not provided in (Wang *et al*, 2000). The P-values of the paired t-tests on HEPA data set as listed in Table 3.10 (EvoC *vs* C4.5: $P = 1.36\times10^{-12}$; EvoC *vs* PART: $P = 5.56\times10^{-10}$; EvoC *vs* Naïve Bayes: $P = 0.31$) show that the EvoC outperforms C4.5 and PART, and is comparable to Naïve Bayes based on the average results over the 100 simulation runs when the level of significance α is set as 0.005.

[2] Recently, the WBCD data set is widely adopted by many machine-learning algorithms in the medical domain. Therefore comparisons between different algorithms based on this data set are relatively more comprehensive than the other two data sets studied in this paper.

Table 3.11 The comparison results for the HEPA data set

Algorithm	# Rules	Time (s)	Average accuracy	Best accuracy	Standard deviation
EvoC	2.93	$4.84\text{x}10^5$	83.92	94.34%	4.03%
C4.5 (Quinlan, 1993)	5.85	< 1	78.94	90.57%	4.84%
PART (Frank and Witten, 1998)	6.64	<1	80.02	94.34%	4.98%
Naïve Bayes (John and Langley, 1995)	-	< 1	83.62	94.34%	4.90%
GA-based FKIF (Wang *et al*, 2000)	-	-	-	92.9%	-

B. Comparison Results for the WDBC Data Set

Table 3.12 compares the results from EvoC, C4.5, PART, and Naïve Bayes for the WDBC data set. It can be seen that the EvoC produces competitive classification accuracies, besides giving the smallest standard deviation among all methods. In addition, the P-values of the paired t-tests on WDBC data set as listed in Table 3.10 (EvoC *vs* C4.5: $P = 3.54 \times 10^{-2}$; EvoC *vs* PART: $P = 2.36 \times 10^{-3}$; EvoC *vs* Naïve Bayes: $P = 2.64 \times 10^{-9}$) show that the EvoC outperforms the algorithms of C4.5, PART and Naïve Bayes based on the average results over the 100 simulation runs when the level of significance α is set as 0.05.

Table 3.12 The comparison results for the WDBC data set

Algorithm	# Rules	Time (s)	Average accuracy	Best accuracy	Standard deviation
EvoC	9.74	$4.50\text{x}10^5$	93.04%	96.37%	1.47%
C4.5 (Quinlan, 1993)	10.06	< 1	92.61%	97.93%	1.98%
PART (Frank and Witten, 1998)	6.23	< 1	92.35%	97.41%	1.65%
Naïve Bayes (John and Langley, 1995)	-	< 1	91.56%	95.37%	2.01%

C. Comparison Results for the WBCD Data Set

Peña-Reyes and Sipper (1999) proposed a fuzzy-genetic approach by combining fuzzy logic and evolutionary algorithms to form a diagnostic system. In the total of 120 evolutionary runs (Peña-Reyes and Sipper, 1999), 78 runs led to fuzzy systems with accuracies exceed 96.5% and 8 runs with accuracies exceed 97.5%. As shown in Table 3.13, only the three best performances of fuzzy-genetic approach are comparable to the average results of 97.57% by EvoC. Moreover, the average performance of fuzzy-genetic approach over the 120 runs is 96.02%, which is only slightly better than the worst rule sets (with a predictive accuracy of 95.26%) generated by EvoC. If only the best results are considered, a four-rule fuzzy system achieves the predictive accuracy of 98.24%, which is lower than the best predictive accuracy of 99.13% by EvoC.

The EvoC has also been compared with the NeuralRule approach proposed by Setiono (2000), which is capable of extracting classification rules from trained neural networks. Setiono (2000) trained 200 neural networks in total and after pruning the network to 95% and 98% accuracies on the training set, an accuracy of 95.44% and 96.66% was achieved on the testing set respectively. In terms of the best results produced by the pruned networks, NeuralRule achieves an accuracy of 98.25% on the testing set, which is lower than the best predictive accuracy of 99.13% by EvoC.

Table 3.13 The comparison results for the WBCD data set

Algorithm	**# Rules**	**Time (s)**	**Average accuracy**	**Best accuracy**	**Standard deviation**
EvoC	5.99	$3.35x10^5$	97.57%	99.13%	0.51%
C4.5 (Quinlan, 1993)	8.99	< 1	95.09%	97.84%	1.16%
PART (Frank and Witten, 1998)	9.03	<1	95.33%	98.28%	1.16%
NaïveBayes (John and Langley, 1995)	-	< 1	96.37%	98.28%	0.89%
NeuroRule-rule 3 (Setiono, 2000)	5	-	-	98.24%	-
Fuzzy-GA4 (Peña-Reyes and Sipper, 1999)	4	-	96.02%	98.24%	-

Although the EvoC is capable of evolving comprehensible classification rules with good generalization performance, it often requires extensive computational effort as compared to existing approaches. The EvoC is generally developed for off-line data classification, which could be useful for many applications where the training time is less important than the generalization in classification. To reduce the computational effort significantly, the EvoC can be integrated into the 'Paladin-DEC' distributed evolutionary computing framework (Tan *et al*, 2002b), where multiple inter-communicating subpopulations will be implemented to share and distribute the classification workload among multiple computers over the Internet.

3.5 Conclusion

A two-phase hybrid evolutionary classifier capable of extracting comprehensible classification rules with good accuracy in medical diagnosis has been proposed in this chapter. In the first phase, genetic programming has been applied to evolve nominal attributes for free structured rules while genetic algorithms have been used to optimize the numeric attributes for concise classification rules without the need of discretization. The second phase then formulates accurate rule sets by optimizing the order and number of rules in the evolution based upon the pool of confined candidate rules obtained in the phase 1. The proposed evolutionary classifier has been validated upon one hepatitis and two breast cancer datasets, which are representative real-world data collected to aid the prognosis and diagnosis of disease. Simulation results show that the EvoC produces comprehensible and good classification rules for the three medical datasets. Results obtained from the t-tests further justify its robustness and invariance to random partition of datasets.

Chapter 4

Distributed coevolution for rule induction

4.1 Introduction

Although evolutionary algorithms have been applied to various applications in data mining, the computation cost involved in terms of time and hardware resources often increases as the size or complexity of the problem becomes larger. One promising approach to overcome this limitation is to exploit the inherent parallelism of evolutionary algorithms by creating an infrastructure necessary to support distributed evolutionary computing using existing Internet and hardware resources. This chapter presents a distributed coevolutionary data mining system (DCDM) for extracting comprehensible rules in data mining, which allows different species to be evolved cooperatively and simultaneously, while the computational workload is shared among multiple computers over the Internet. Through the inter-communications among different species of rules and rule sets in a distributed computing approach, the concurrent processing and computational speed of the coevolutionary classifier are enhanced significantly. The advantages and performance of the proposed DCDM are extensively validated upon various datasets obtained from UCI machine learning repository. It is shown that the predicting accuracy of the DCDM is robust and the computational time is substantially reduced as the number of remote engines increases. Comparison results illustrate that the DCDM produces comprehensible and good classification rules for all the datasets, which are very competitive as compared with existing classifiers in literature.

In fact, exploiting the intrinsic parallelism of EAs, various parallel evolutionary algorithms have already been proposed to reduce the computational effort needed in solving complex optimization problems (Cantú-Paz, 1998; Cristea and Godza, 2000; Nang and Matsuo, 1994; Tan *et al*., 2002a). Instead of evolving the entire population in a single processor, the parallel evolutionary algorithms applied the concept of multiple inter-communicating subpopulations (Cristea and Godza, 2000) in analogy with the natural evolution of spatially distributed populations. Such inter-communication allows individuals to migrate among the subpopulations based upon some patterns to induce diversity of elite individuals periodically, in a way that simulates the species evolved in natural environment. These parallel evolutionary algorithms have been applied to solve many sophisticated problems in various fields, such as image processing (Chen *et al*., 1996), VLSI (Yoshida and Yasuoka, 1999), network design (Sleem *et al*., 2000), and drug scheduling (Tan *et al*., 2002a).

In DCDM, two species namely rules (fundamental elements) and rule sets (complex elements) are evolving simultaneously and cooperatively. Rule population and a number of rule set populations are distributed among multiple computers over Internet. These coevolving populations are coupled cooperatively on fitness as the quality of rule sets greatly depends on the quality of rules forming the rule sets. Hence, through the inter-communications between the rule population and the rule set populations, rules thus generated are all crucial to the problem and useful for rule set construction, which makes it easy to find the resultant rule set with a fairly good performance

4.2 The framework of DCDM

The framework of the DCDM is presented in Figure 4.1 to give an overall impression of the entire system. The system is built upon the foundation of Java technology offered by Sun Microsystems and is equipped with application programming interfaces (APIs) and technologies from J2SE. The infrastructure of the distributed framework is the Java™ Remote Method Invocation (RMI) system, which allows an object running in one Java Virtual Machine (VM) to invoke methods on an object running in another Java VM. RMI provides for remote communication between programs written in the Java programming language. RMI applications are often comprised of two separate programs: a server and a client. A typical server application creates some remote objects, makes references to them accessible, and waits for clients to invoke methods on these remote objects. A typical client application gets a remote reference to one or more remote objects in the server (remote compute engine) and then invokes methods on them. RMI provides the mechanism by which the remote compute engine and the client communicate and pass information back and forth.

The overall system is made up of a client and a number of remote compute engines. The client represents a data-mining task, which is in charge of identifying the remote compute engine, dispatching the workload and collecting the final result. Besides these, in DCDM, client is also a major computational component because the rule population is also evolved on it. Ordinarily, all the datasets are garnered in a central database, which can be accessed by the client and all the remote compute engines. When the client starts, it first collects the information of the available compute engines from an HTTP server and then broadcasts to its selected engines on which dataset they should work. The engine information is uploaded by itself when the engine registers in the sever registry from which the client can locate all the engines. After the engine registers itself successfully, it begins to wait the client to assign the task. Once a task is assigned, the engine will read the information from the client and extract class name and path before loading the class remotely from the server. If the class loaded is consistent with the specification of the system, the computation procedure will be initiated. In Figure 4.1, the left part illustrates the situation of client while the right part depicts the remote engines. Communications between the above two parts are resorted to the Java™ Remote Method Invocation (RMI) bridge. The design and implementation details of DCDM are given in the following subsections.

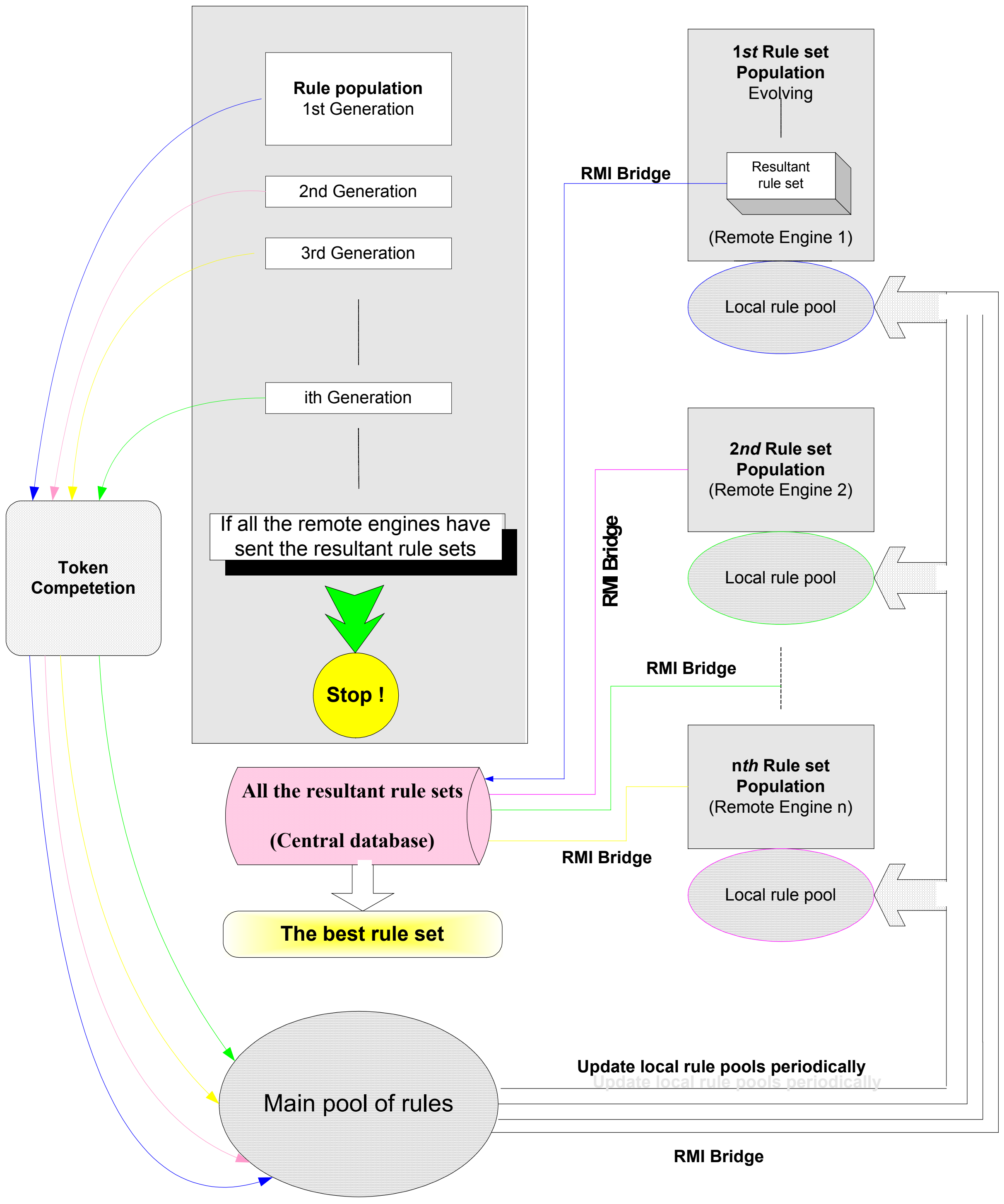
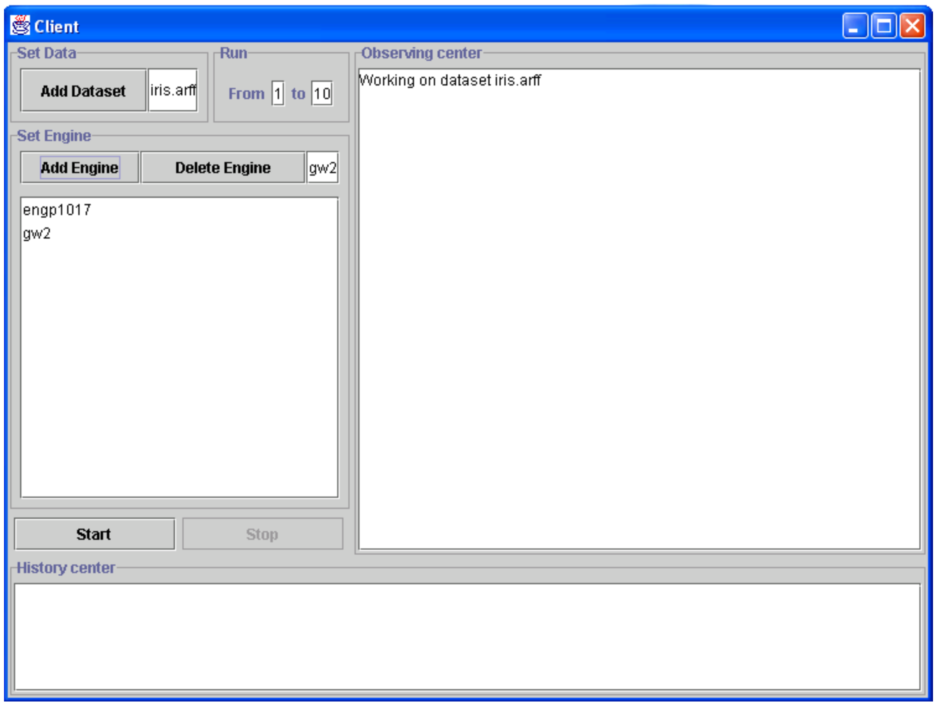


Figure 4.1: Framework of DCDM system

4.3 Client-side design

As shown in the client user interface in Figure 4.2, working dataset is first selected and broadcasted to all the registered remote engines. Before starting evolving the rule population, users can freely choose among the available remote engines to dispatch their workloads. In addition, to evaluate the average performance of the evolutionary algorithm, the running times may also be specified in the client panel. The rule evolving process is launched in the client after all the necessary preparation. As shown in Figure 4.1, the rule evolving would be carried out generation by generation and not stopped until all the resultant rule sets are sent back by the remote compute engines. During the course of rule evolution, a main rule pool is set up in the client and then maintained by being fed with the rules from the rule population. The following sections detail the process of rule evolution.



iris.arff

1

10

Working on dataset iris.arff

gw2

engp1017
gw2

Figure 4.2: The client user interface

In the rule population, chromosomes are encoded using Michigan approach where each chromosome represents a single rule. These chromosomes are variable in length, and all the initial chromosomes are evaluated against the training dataset for their fitness before starting the iteration looping. The fitness function is the same as the one applied in EvoC of Chapter 3. The mating pool is first formed by selecting parents from the rule population using tournament selection. The genetic operators such as crossover and mutation are then applied upon the mating pool to reproduce the offspring. The offspring are assigned as the new main population and passed into the token competition that works as a covering algorithm. The token competition effectively maintains a pool of good rules, i.e., rules that covers the solution space well. As classification problems generally contain not only one but many useful knowledge or regulations, it is crucial for the coevolutionary algorithm to maintain a population with high diversity. To achieve this, a regenerate operator is used, which replaces chromosomes that are below average fitness in the main population with randomly generated chromosomes at some user specified probability. After the regeneration, all chromosomes in the pool resulting from the token competition are used to form the main rule pool, which serves as the resource to build the local rule pool for every rule set population. The evolution of the rule population will not be stopped until all the rule set populations have finish their evolutions.

4.4 Engine-side design

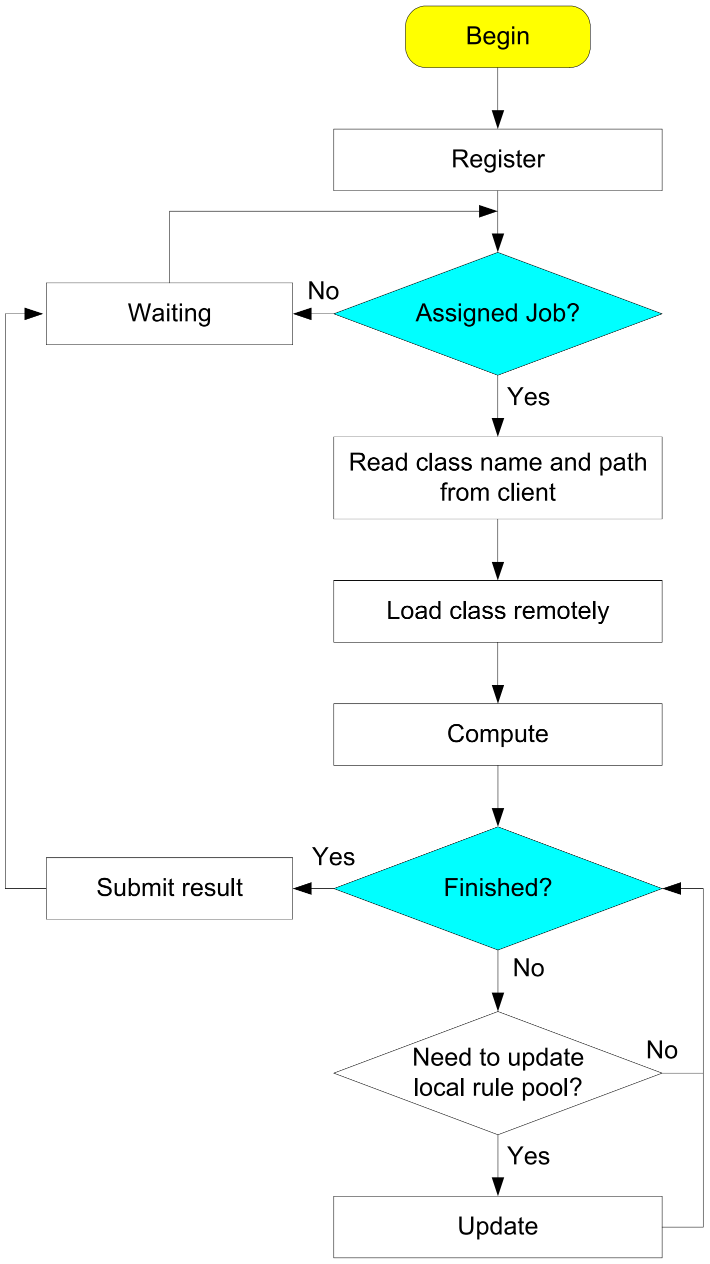


Figure 4.3: The working process of a remote engine

As stated in the previous section, rule evolution is conducted in the client side, while in the remote engine side, the final result namely the rule set is evolved. The working process of a remote engine is shown in Figure 4.3. The DCDM algorithm applies a group of rule set populations to evolve rule sets with different number of rules. The chromosomes in these populations are encoded with Pittsburgh approach where each chromosome is encoded with a rule set. The basic element that builds up these chromosomes is the rules from their respective local rule pools, as shown in Figure 4.1. The number of rules in a chromosome depends on which rule set population the chromosome is attached to. For example, the fourth rule set population will only contain chromosomes with 4 rules. Note that the default class is also encoded in the chromosome, which provides greater flexibility on constructing the rule sets. The number of rule set populations is determined by the maximum number of rules allowed in a rule set. For example, if a rule set is allowed to have up to 15 rules, then there will be 15 such populations.

The quality of these rule sets is greatly affected by the rules used. To evaluate the chromosomes, the classification accuracy on the training set is used. Here, only mutation operator is applied to evolve the chromosomes in order to avoid the reproduction of redundant rule sets. At the end of the evolution, each rule set population outputs its 'best' candidate rule set, which will be gathered together with the best rule sets from other populations. Competition will then be performed between these rule sets to obtain the final optimal one. To retain concise rule sets in the classification, a shorter rule set is preferable to a longer one even if both achieved the same classification accuracy. In this way, the order and number of rules in the rule sets can be optimized and determined simultaneously.

4.5 Update of the local rule pools

In the client side, after the rule evolution finishes its first generation, the main rule pool is built up from the rule population with the assistance of token competition. While in the remote engine side, the rule set populations can still not be initialized until their respective local rule pools are set up. So a user specified portion of rules will be chosen randomly from the main rule pool to initialize the local rule pools on the remote engines and afterwards these local rule pools will be updated periodically by the new rules from the main rule pool. The updates of local rule pools on different remote engines are not carried out simultaneously because communications between client and remote engines are conducted individually. These local rule pools are served like the factory of raw materials for the rule set evolution. After the initialization of their respective local rule pools, the remote engines launch the rule set evolution. Since the local rule pools on different remote engines are not identical, rules from different rule set populations are also different, which might carry more information due to variety. The evolution of the rule sets will not be stopped until reaching a predefined generation number. In the end, the rule set with the best training accuracy will be sent back to a central database. After colleting all the resultant rule sets from all of the remote engines, the client is in charge of making a selection among them based on their competition results.

4.6 Distribution of the workload

A task queue is resorted to distribute the rule set populations to the available remote compute engines. Figure 4.4 illustrates how this task queue can help this work done. Suppose there are 15 rule set populations and 4 remote compute engines available. Each of the 15 rule set populations will be assigned with a task ID, for instance, the first rule set population will be regarded as TASK1, so on and so far. At the very beginning, the 15 tasks all enter the queue and the first 4 of them will be assigned to the 4 remote compute engines. Together with the task IDs, a portion of the rules from the main rule pools also migrate to the compute engines to form the local rule pools. So the task ID tells what kind of rule sets the compute engine should build up. After the evolution is finished, the best rule set is sent to and stored in a central database. At the same time, the compute engine, which finishes the task, sends a message indicating that it is free and the task ID on the top of the queue will be assigned to it once the message has been caught. If problems happened on the remote engine's side, which make the tasks cannot be finished successfully, an exception is invoked to inform the failure of the tasks. In this case, the failed task ID reenters the task queue and will be assigned to available compute engine at its turn.

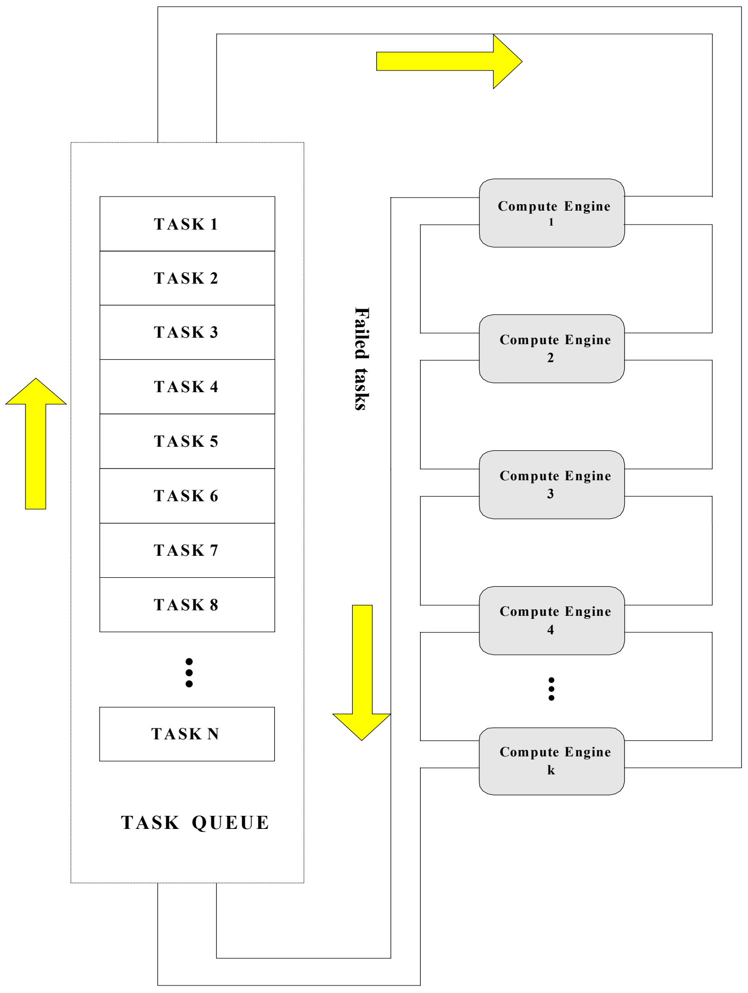


Figure 4.4: Distribution of the workload

4.7 Workload Balancing

Since the processing power and specification for various computers in a network might be different, the feature of work balancing that ensures the remote engines are processed in a similar pace is needed in a distributed evolutionary system. This is important because the total computational time is decided by the remote engine, which finished its work last and if the remote compute engine with the least computational capacity is assigned the most heavy workload, not only would longer time be required but the resources would be greatly wasted. Intuitively, work balancing for a distributed evolutionary system could be difficult due to the fact that the working environment in a network is often complex and uncertain. DCDM resorts to a simple work balancing strategy by assigning the workload to the remote compute engines according to their respective computational capacities. Recalling the content of section 4.2, one could remember that when a remote engine is first launched, it uploads its configuration information to a HTTP server, which can be accessed by the client. The hardware configuration of the remote engine is recorded in the information file such as the CPU speed, RAM size, etc. Reading the information file from the HTTP server, the client carries out a simple task scheduling and assigns different tasks to respective remote engines mainly according to their computational capacities.

4.8 Experimental studies

4.8.1 Experimental setup

Table 4.1 lists the parameter settings in DCDM that are applied to all the testing problems. These parameters have been chosen after some preliminary experiments and then applied upon all the experiments. Therefore the settings should not be regarded as an optimal set of parameter values but rather a generalized one with which the DCDM can perform well over a wide range of datasets. The DCDM was programmed using the Java Developers Kit (JDK 1.4.1) from Sun Microsystems. The rule population is evolving on an Intel Pentium IV 1.3 GHz computer with 128 MB SDRAM. Four computers serve as the remote compute engines and their configurations are listed in Table 4.2.

Table 4.1 Parameter settings used in the experiments

Parameter	Value
Population size	100
Co-population size	50
Number of generations	100
Number of co-populations	15
Probability of crossover	0.9
Probability of mutation	0.3
Probability of regeneration	0.5

Table 4.2 Configurations for the remote compute engines

Engine number	Configuration (CPU (MHz)/RAM (MB))
1	PIII 800/ 512
2	PIII 933/512
3	PIV 1300/ 128
4	PIII 933/ 256

The proposed DCDM is validated based on 6 datasets, which are made up of the iris dataset, 4 medical datasets and a credit card assessment dataset. Each of these datasets is partitioned into two sets: a training set and a testing set (also called validation set). The training set is used to train DCDM, through which its learning capability can be justified. However, a classifier that learns well does not necessarily guarantee it is also good in generalization. In order to evaluate the generalization capability, the rule sets obtained by DCDM are applied to testing set after the training. In order to ensure the replicability and clarity of the validation results, all experiments have been designed carefully in this study.

In the total of 100 evolutionary runs on each of the 6 datasets, a random seed[3], which is the same as the number of runs (i.e. the 50th run uses random seed 50), is first used to randomize the orders of data in the datasets. The randomized data is then partitioned with the first 66% as the training data and the remaining 34% as the test data.

4.8.2 The problems sets

The datasets used to validate the performance of DCDM are obtained from UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). All these datasets are careful chosen with many considerations such as the number and type of the attributes (nominal, numeric or both), containing the missing attribute values or not, number of data samples, number of classes and the ratio of majority class to minority class etc. Table 4.3 describes the domains of the data together with the classification tasks, while in Table 4.4 the characteristics of each dataset are given in detail.

Table 4.3 Classification task descriptions of the datasets

Dataset	Domain	Classification task
Iris	Botany	Classify 3 species of iris flower based on their physical characteristic.
Breast cancer	Medicine	Determine the patients for whom the cancer will re-occur.
Heart-c	Medicine	Determine the risk of heart disease given certain medical conditions in patients.
Diabetes	Medicine	Determine whether a patient shows signs of diabetes according to World Health Organization criteria.
Hepatitis	Medicine	Determine whether a hepatitis patient will live or die according to given medical conditions.
Credit-a	Finance	Determine a certain aspect of credit card applications given other specifications.

[3] The random number generator used in the experiments is provided with Sun's JDK 1.4.1 and the data set randomizer used is provided with WEKA (Witten and Frank, 1999). Different partitioning of data sets might have resulted under different programming environments.

Table 4.4 The characteristics of the datasets

Dataset	Number of attributes	Number of classes	Number of instances	Percentage of major class	Attribute characteristics		
					Numeric	Nominal	Missing
Iris	4	3	150	33	Yes	No	No
Breast cancer	9	2	286	66	No	Yes	Yes
Heart-c	13	5	303	55	Yes	Yes	Yes
Diabetes	8	2	768	65	Yes	No	No
Hepatitis	19	2	155	79	Yes	Yes	Yes
Credit-a	15	2	690	56	Yes	Yes	Yes

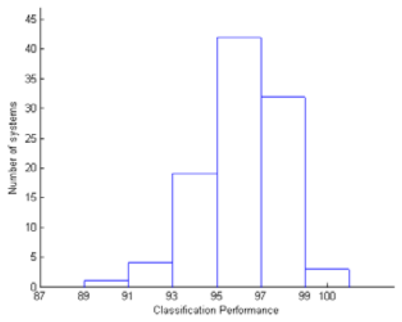
4.8.3 Experimental results

Table 4.5 summarizes the classification results produced by DCDM over the 100 independent runs for all the testing problems. The statistics in the table could firstly give a general impression that DCDM generates classification rule sets with a fairly high predictive accuracy, a stable performance (reflected by the small standard deviation) and a small number of rules. The histograms in Figure 4.5 illustrate the detailed results of DCDM over 100 runs, which generally show a normally distributed performance. The best rule sets of all the 6 datasets are presented in the Table 4.6 ~4.11, from which one can see what the resultant rule sets look like.

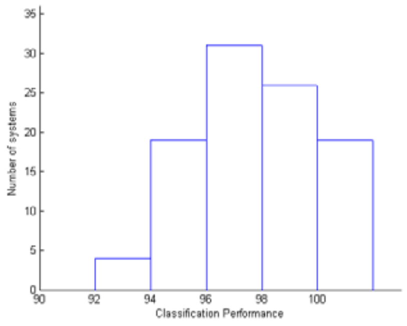
Table 4.5 Classification results from DCDM

DCDM	Iris	Breast cancer	Heart-c	Diabetes	Hepatitis	Credit-a
Training						
Max	100%	80.32%	84.42%	79.05%	93.14%	88.35%
Mean	96.53%	76.30%	79.72%	75.79%	75.99%	86.00%
StdDev*	1.80%	1.79%	2.70%	1.27%	2.54%	1.03%
Testing						
Max	100%	84.39%	86.54%	79.77%	92.45%	90.21%
Mean	96.73%	76.16%	80.01%	75.31%	84.38%	86.28%
StdDev	2.40%	3.11%	3.19%	2.31%	3.81%	1.79%
Length*	4	5	8	6	5	4
Training time (s)	56.77	63.81	98.32	124.44	63.66	153.22

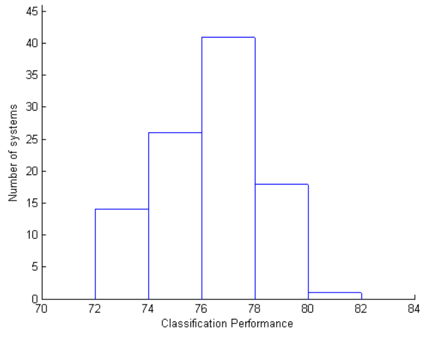
*StdDev represents the standard deviation and length represents the number of rules in a rule set.



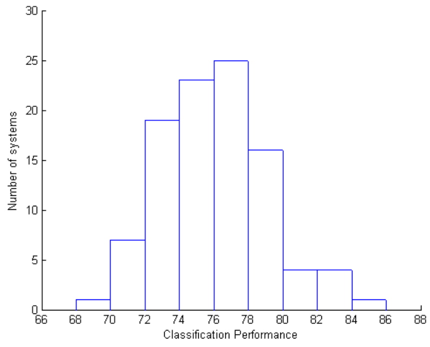
Iris (train)



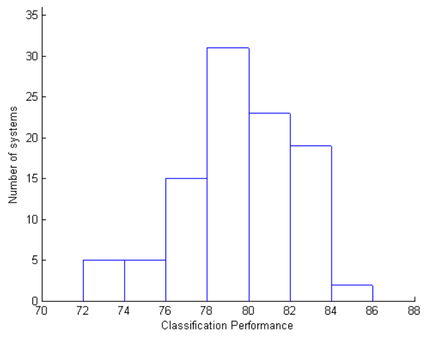
Iris (test)



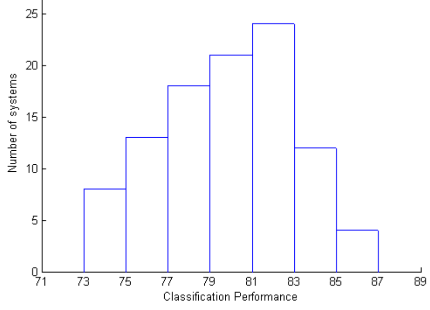
Breast-cancer (train)



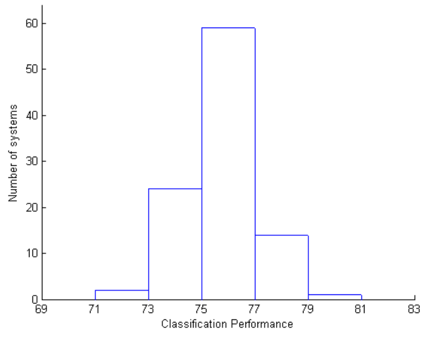
Breast cancer (train)



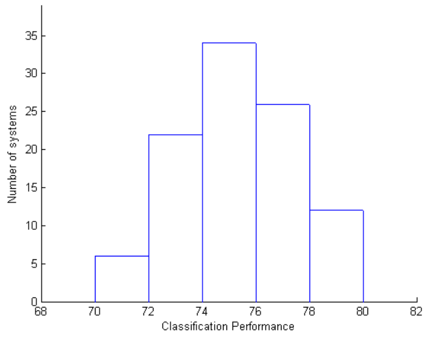
Heart-c (train)



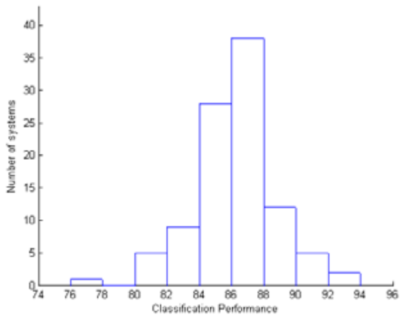
Heart-c (test)



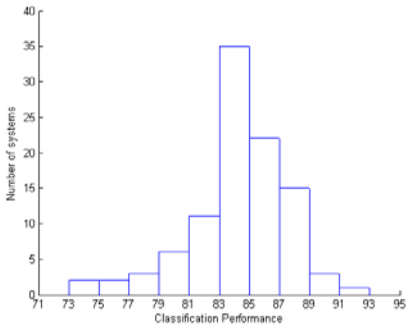
Diabetes (train)



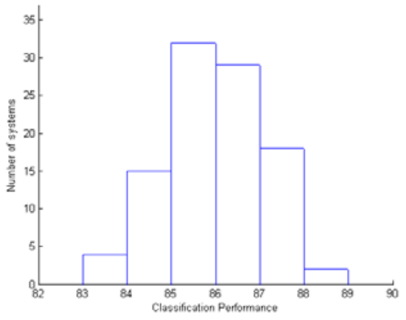
Diabetes (test)



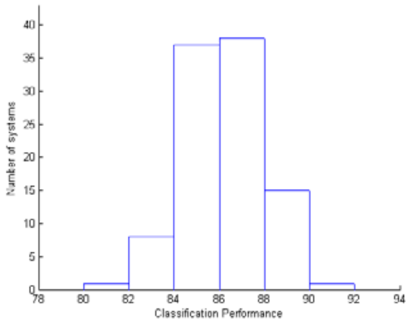
Hepatitis (train)



Hepatitis (test)



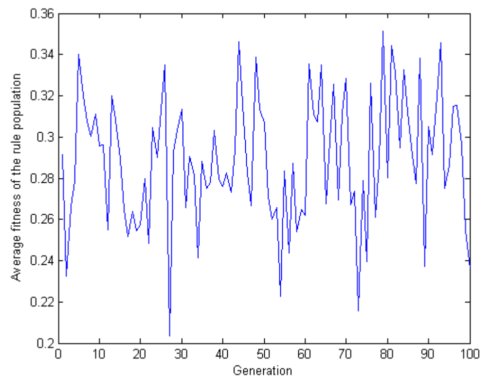
Credit-a (train)



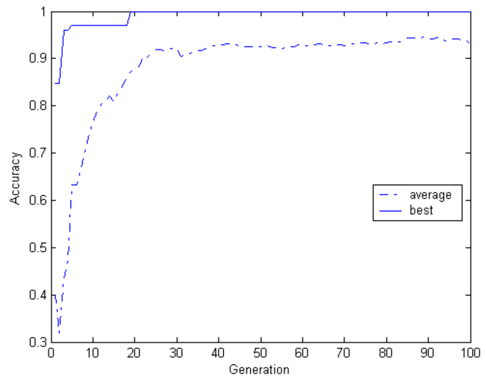
Credit-a (test)

Figure 4.5: Diagrams of the classification results

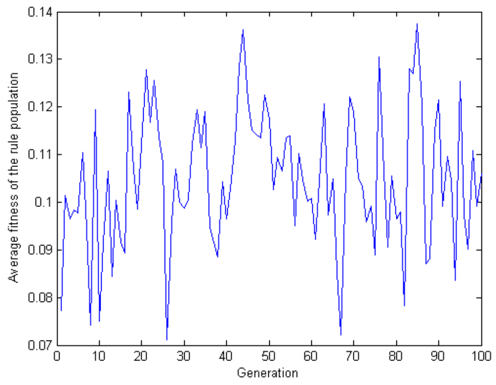
Figure 4.6 shows the convergence performance of DCDM for all the datasets. The figures on the left side are the average fitness of the rule population while the right-side ones are average as well as the best accuracy (also fitness) over all the rule set populations. As can be seen, although the rule population evolves in a very stochastic way (mainly due to the regenerating operator), it provides a ground for the rule set populations to progress in a positive direction and resulted in a good exponentially increased convergence trace. This shows how the populations are coevolved cooperatively to produce the good solutions. The stochastic nature of the rule population plays an important role in the proposed coevolutionary model to maintain the diversity of the individual pool.



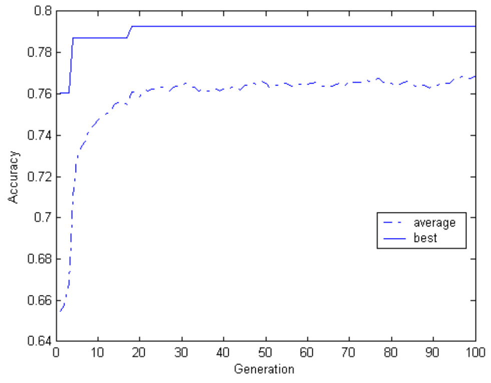
Iris (rule population)



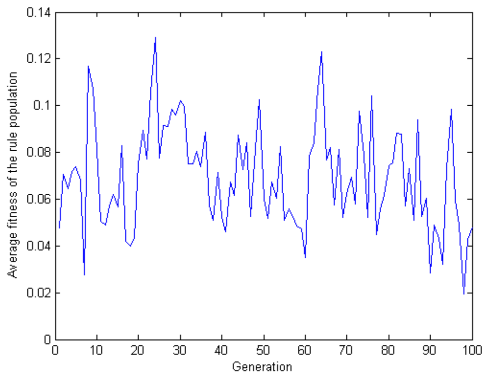
Iris (rule set populations)



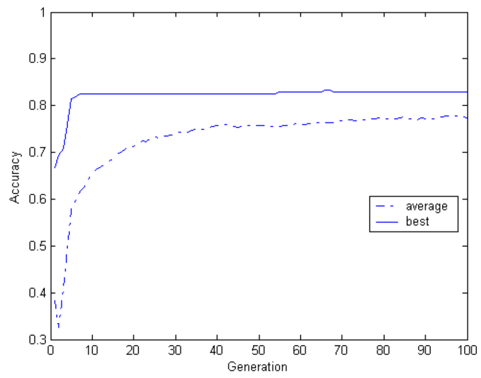
Breast cancer (rule population)



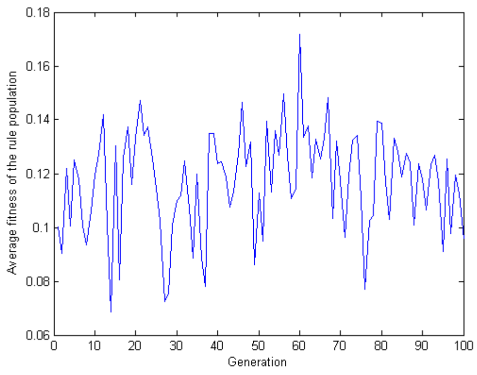
Breast cancer (rule set populations)



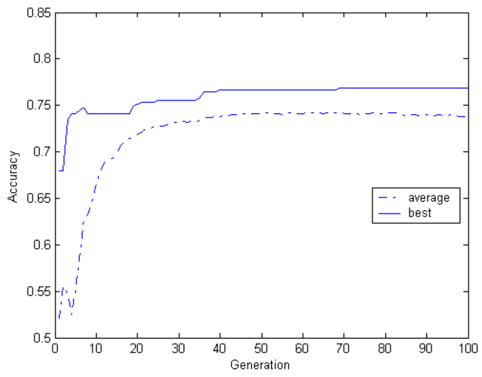
Heart-c (rule population)



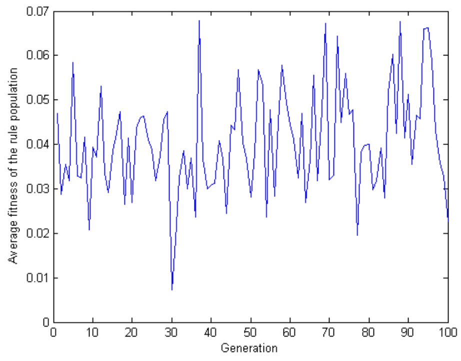
Heart-c (rule set populations)



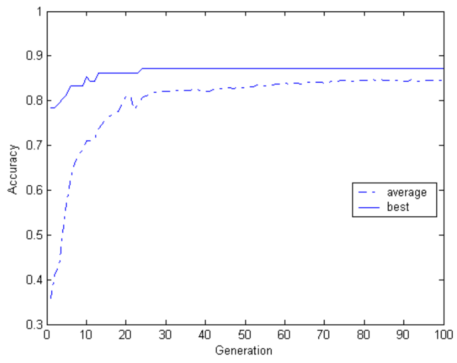
Diabetes (rule population)



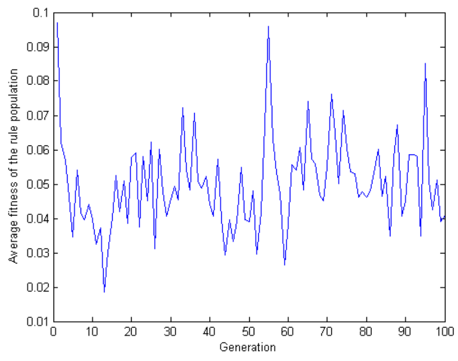
Diabetes (rule set populations)



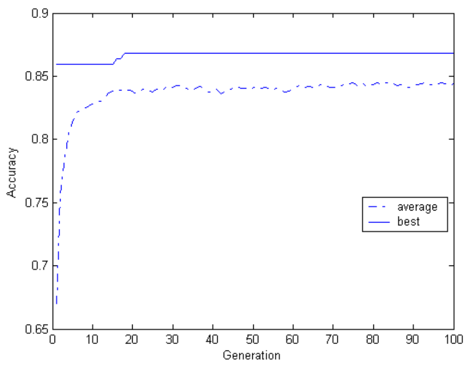
Hepatitis (rule population)



Hepatitis (rule set populations)



Credit-a (rule population)



Credit-a (rule set populations)

Figure 4.6: Evolutionary progress on the rule and rule set populations

Table 4.6 The best classification rule set of DCDM for the Iris dataset

	Rule	**Fitness**	**Support factor**	**Confidence factor**

1	IF petallength < 2.5419,
	THEN **class** = Iris-setosa
		2.0	0.3232	1.0

2	IF petalwidth <> [0.1, 1.6807],
	THEN **class** = Iris-virginica
		1.8735	0.3333	0.9429

3	IF petallength < 4.9878
	AND petalwidth > 0.8904,
	THEN **class** = Iris-versicolor
		1.6909	0.3131	0.8378

4	IF petallength >= 4.6518,
	THEN **class** = Iris-virginica
		1.6562	0.3333	0.7857

5	ELSE **class** = Iris-setosa			
Accuracy = 100%				

Table 4.7 The best classification rule set of DCDM for the Breast Cancer dataset

	Rule	**Fitness**	**Support factor**	**Confidence factor**

1	IF menopause != lt40 ,
	AND deg-malig != 3 ,
	THEN **class** = no-recurrence-events
		1.0441	0.5372	0.7594

2	IF node-caps = yes ,
	THEN **class** = no-recurrence-events
		0.9478	0.5691	0.7181

3	IF inv-nodes = 0-2,
	THEN **class** = no-recurrence-events
		0.9924	0.5426	0.7391

4	IF inv-nodes != 24-26,
	AND node-caps!= yes ,
	THEN **class** = no-recurrence-events+
		0.6985	0.3351	0.3663

5	ELSE **class** = recurrence-events			
Accuracy = 84.69%				

Table 4.8 The best classification rule set of DCDM for the Heart-C dataset

	Rule	**Fitness**	**Support factor**	**Confidence factor**

1	IF exang = no
	AND ca < 0.5106,
	THEN **num** = <50
		1.2361	0.3769	0.8152

2	IF cp = asympt ,
	AND chol >= 156.3957,
	THEN **num** = >50_1
		1.109	0.3467	0.734

3	IF thal = normal ,
	THEN **num** = <50
		1.1414	0.3819	0.7451

4	ELSE **num** = >50_1			
Accuracy = 86.54%				

Table 4.9 The best classification rule set of DCDM for the Diabetes dataset

	Rule	Fitness	Support factor	Confidence factor

1	IF plas >< [0, 126.267],
	AND pres <> [0, 42.5834],
	THEN **class** = tested_negative
		1.0551	0.4644	0.7807

2	IF plas > 93.0716,
	AND mass > 29.7921,
	THEN **class** = tested_positive
		1.0394	0.2826	0.5500

3	IF preg >< [0, 14.4087],
	AND plas >< [0, 148.2526],
	AND insu <= 817.285,
	THEN **class** = tested_negative
		1.0497	0.5929	0.7264

4	IF plas >< [0, 152.2424],
	ANDmass <> [0, 16.3087],
	THEN **class** = tested_negative
		1.0230	0.5909	0.7188

5	IF age > 28.4661,
	THEN **class** = tested_negative
		0.9684	0.2905	0.5034

6	IF mass >= 25.6667,
	THEN **class** = tested_positive
		0.8278	0.3458	0.4258

7	ELSE **class** = tested_negative			
Accuracy = 79.77%				

Table 4.10 The best classification rule set of DCDM for the Hepatitis dataset

	Rule	Fitness	Support factor	Confidence factor

1	IF SPLEEN_PALPABLE != yes ,
	AND ASCITES != yes ,
	THEN **class** = LIVE
		1.3256	0.6471	0.9041

2	IF FATIGUE = yes ,
	AND ALBUMIN >< [2.1, 3.669],
	THEN **class** = DIE
		1.0784	0.1471	0.5556

3	IF LIVER_FIRM != no,
	AND ASCITES = no ,
	AND BILIRUBIN <= 3.1185,
	THEN **class** = LIVE
		1.2911	0.6765	0.8846

4	IF ALBUMIN > 3.3452,
	THEN **class** = LIVE
		1.2855	0.6275	0.9014

5	IF ALBUMIN >= 3.0876,
	THEN **class** = LIVE
		1.2799	0.6471	0.8919

6	IF ANTIVIRALS = no ,
	AND HISTOLOGY != no ,
	THEN **class** = DIE
		1.0582	0.1765	0.4186

7	IF ASCITES = no ,
	AND ALBUMIN <= 5.9173,
	THEN **class** = LIVE
		1.2350	0.6471	0.8800

8	IF SPIDERS != yes ,
	THEN **class** = LIVE
		1.1636	0.5882	0.8824

9	ELSE **class** = DIE			
Accuracy = 92.45%				

Table 4.11 The best classification rule set of DCDM for the Credit-A dataset

	Rule	Fitness	Support factor	Confidence factor

1	IF A9 = f,
	THEN **class** = -
		1.4177	0.4176	0.9268

2	IF A10 != f ,
	THEN **class** = +
		1.0370	0.2967	0.7031

3	IF A5 = g,
	AND A8 >< [0, 18.7653],
	AND A10 = f,
	THEN **class** = +
		0.9413	0.2571	0.7178

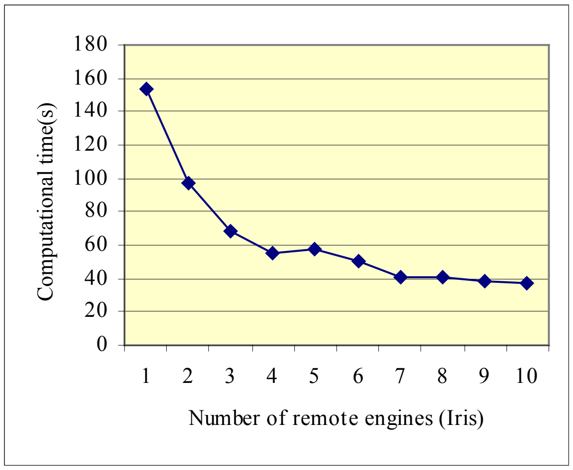
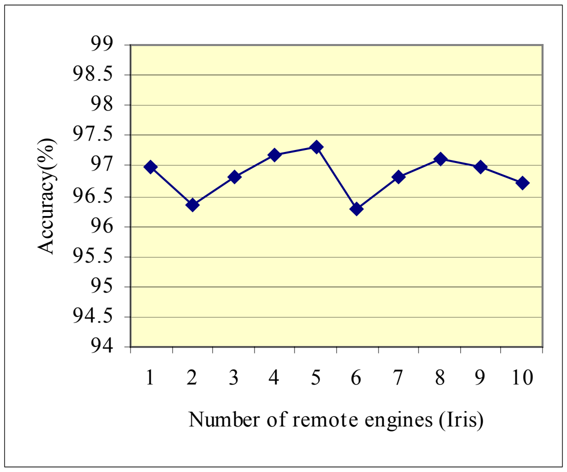
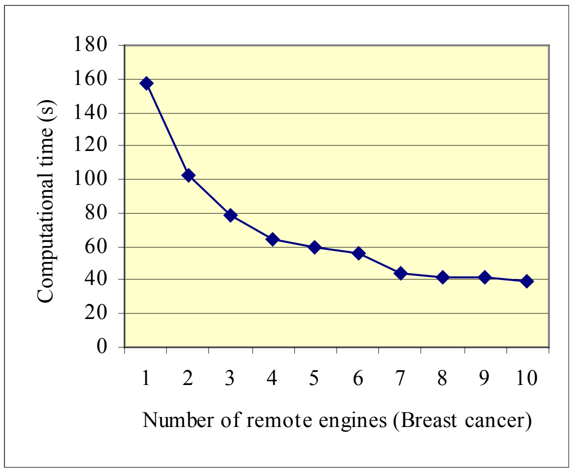
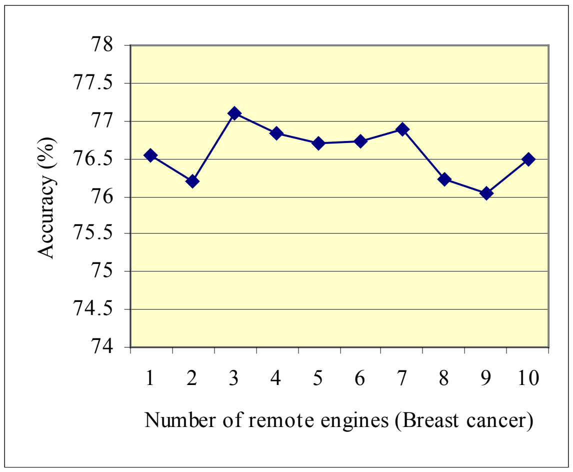
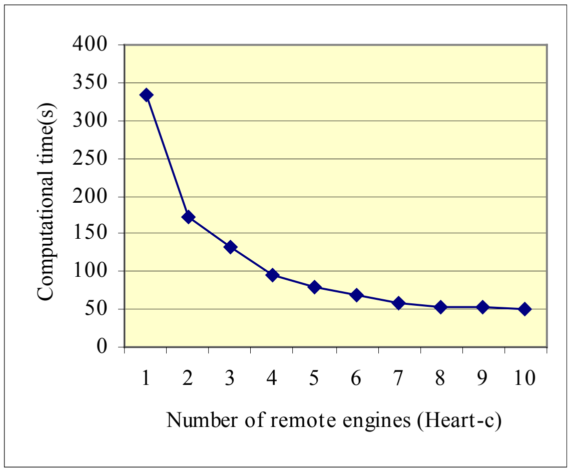
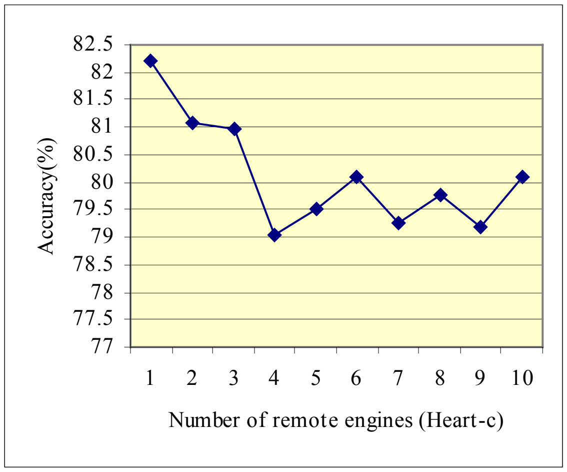
4	IF A1 = b, AND A2 <= 54.192, AND A5 != p, AND A11 >< [0, 33.6464], AND A13 = g , AND A15 >< [0, 74832.9761], THEN **class** = +	0.6423	0.2132	0.5132
5	IF A1 != a , AND A2 >< [13.75, 55.9752], AND A8 >< [0,28.2914], AND A11 >< [0, 39.451], THEN **class** = -	0.7059	0.3670	0.5604
6	ELSE **class** = +			
Accuracy = 90.21%				

4.8.4 Performance analysis

To make a more thorough study of the efficacy of the distributed techniques on both the computational time and the classification accuracy, DCDM is applied to all the testing problems for different number of remote engines ranging from 1 to 10 with a uniform parameter setting as described in Table 4.1. 30 evolution runs are carried out in each case and the average results are summarized in Table 4.12 and Figure 4.7.

Table 4.12 Average results for different number of remote engines

Engine number	Iris	Breast cancer	Heart-c	Diabetes	Hepatitis	Credit-a
Computational time (s)						
1	153.2	156.82	333.39	323.75	155.73	375.89
2	96.62	102.59	171.81	202.33	96.52	254.79
3	67.88	78.5	131.25	146.33	76.67	188.69
4	55.61	64.79	96.49	122.11	64.74	154.6
5	57.34	59.5	80.40	103.91	55.66	131.8
6	50.32	56.16	68.25	101.82	50.83	124.39
7	41.28	44.29	59.33	99.41	44.19	100.64
8	40.99	42.27	53.09	74.99	42.28	99.96
9	39	41.53	51.85	73.6	39.24	94.45
10	36.73	39.64	50.14	70.65	37.64	88.07
Accuracy (%)						
1	96.99	76.53	82.21	75.79	83.9	86.1
2	96.34	76.19	81.09	75.93	84.28	86.07
3	96.8	77.11	80.96	75.75	83.96	86.01
4	97.19	76.84	79.04	75.99	83.96	86.04
5	97.32	76.7	79.52	75.7	83.84	86.16
6	96.27	76.73	80.1	75.17	83.77	86.16
7	96.8	76.9	79.26	75.39	83.52	86.07
8	97.12	76.22	79.78	75.44	84.09	86.07
9	96.99	76.05	79.17	75.41	84.15	86.14
10	96.73	76.5	80.1	75.28	83.3	85.89


Computational time(s)
Number of remote engines (Iris)

Accuracy(%)
Number of remote engines (Iris)

Computational time (s)
Number of remote engines (Breast cancer)

Accuracy (%)
Number of remote engines (Breast cancer)

Computational time(s)
Number of remote engines (Heart-c)

Accuracy(%)
Number of remote engines (Heart-c)

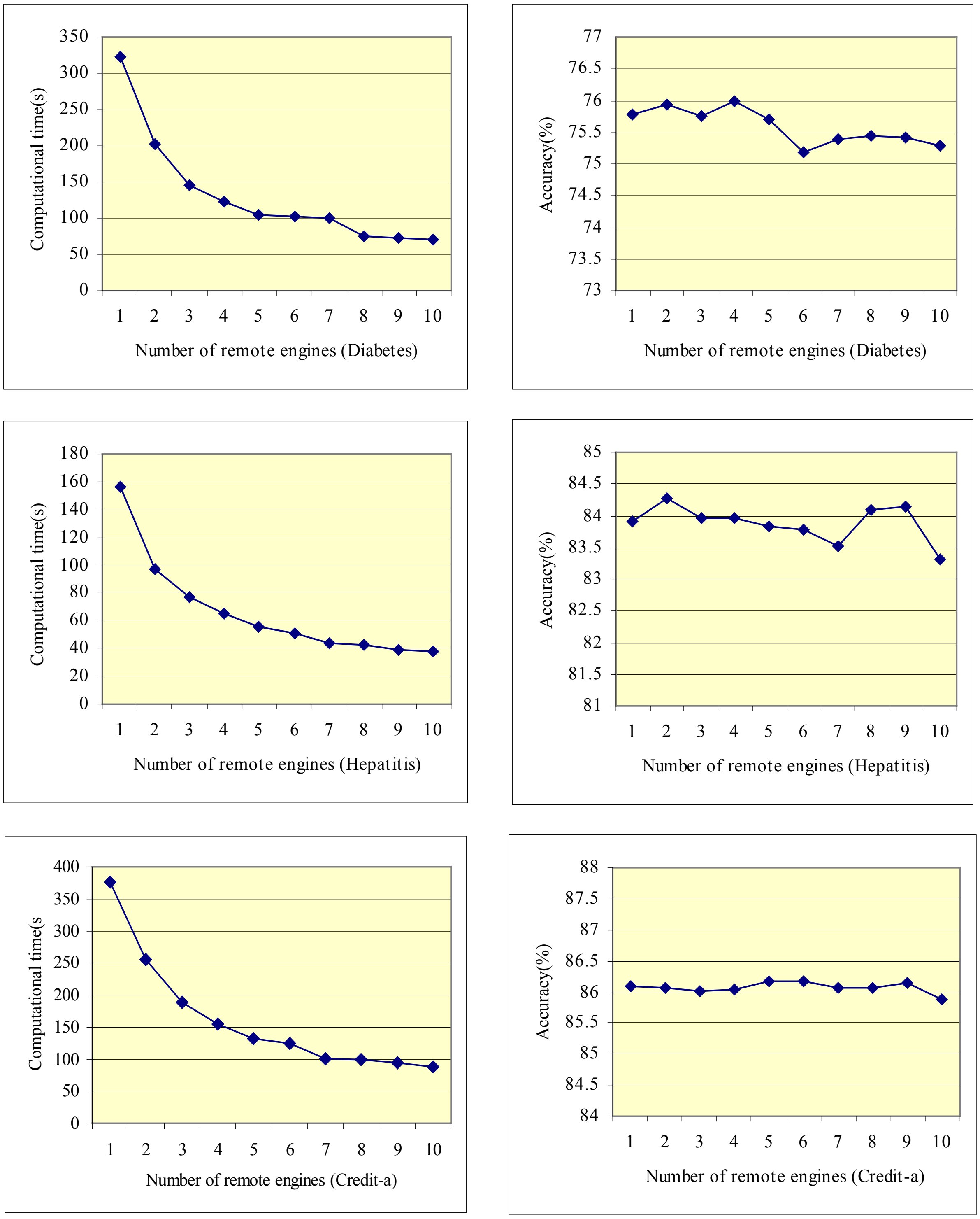


Figure 4.7: Computational time *VS* number of remote engines

As shown in Table 4.12 and Figure 4.7, with different number of remote engines, the computational time for each testing problem varies significantly whereas the changes of the classification accuracy are not that obvious and most of them only fluctuate within 1%. Although the communication status and speed of the Internet often vary from time to time, which may cause minor fluctuation in terms of computational time for a distributed system occasionally, the introduction of the distributed technology into the coevolutionary algorithm has generally greatly improved the efficiency of the overall system. DCDM with ten remote compute engines is on average 5 times faster than the system with only one remote engine. Another observation of Figure 4.7 is that the speed of reduction on computational time decreases with the increment of the number of remote engines, which means that the efficiency improvement will be not obvious when more and more remote engines are added into the system. Four remote engines are chosen in the previous section to present the result is due the reason that when more than 4 remote engines are added into the system the time reduction speed becomes significantly slower, so in the 4-engine situation the Internet resources are most efficiently utilized. While on the other side, because the parameter settings are kept unchanged in DCDM with different remote engines, the results in terms of classification accuracy are also expected not to change much, which can be justified by most of the results of the testing problems. One exception is the Heart-c problem, where there is a 2% fluctuation in accuracy. This can be explained that when the evolution of the rule set is much faster than evolution of the rules, rules that are used to assemble the rule sets are still not good enough, which might impair the performance of the rule set as a whole. While for the other testing problems, the fluctuation is not that much because good rules have already been generated before the rule set evolution is finished, but this situation might also change if more and more remote engines are added into the system.

4.9 Comparisons with other works

4.9.1 Comparisons with three classical machine learning algorithms

It can be seen from the previous section that the proposed DCDM is capable of generating comprehensible rule sets with good classification accuracy. For comparisons, the three machine-learning algorithms, i.e., *C4.5*, *PART* and *Naïve Bayes* described in Chapter 3 have also been applied to the 6 datasets.

Results from the three algorithms are all summarized in Table 4.13. A preliminary comparison can be made by just examining the results from both Table 4.5 and Table 4.13. Generally, DCDM generates concise rule sets with higher classification accuracy and lower standard deviation. To make the comparison a more convincing one, box plot and paired t-test are also conducted in the following parts.

Table 4.13 Results from three classical algorithms

Dataset	**C4.5**			**PART**			**Naïve Bayes**	
	Accuracy	Length	StdDev	Accuracy	Length	StdDev	Accuracy	StdDev
Iris	93.67%	4	3.92%	93.39%	4	3.93%	96.72%	2.93%
Breast cancer	71.81%	14	3.55%	69.32%	14	4.33%	72.34%	3.29%
Heart-c	76.61%	15	4.60%	77.97%	18	4.65%	82.96%	3.37%
Diabetes	73.13%	19	2.55%	72.78%	8	2.59%	75.08%	2.53%
Hepatitis	78.94%	6	4.84%	80.02%	7	4.98%	83.62%	4.90%
Credit-a	85.14%	19	2.06%	83.59%	27	6.98%	77.45%	2.57%

Box plots: As shown in Figure 4.8, the simulation results for all the 6 datasets are represented in the format of box plots (Chambers *et al*., 1983) to visualize the distribution of results in term of classification accuracy over the 100 independent runs. Each box plot represents the distribution of a sample population where a thick horizontal line within the box encodes the median, while the upper and lower ends of the box are the upper and lower quartiles. Dashed appendages illustrate the spread and shape of the distribution, and the '+' represents the outside values. In each graph, the sequence of box plots from left to right is based upon the indexes provide before. As one can see that if the comparison objectives are restricted to the rule-based classifiers, the classification capability of DCDM averaged over 100 runs outperforms the other two in all the datasets. What's more, in 4 of them, DCDM also gives the highest classification accuracy. If Naïve Bayes is also included in the comparisons, it outweighs DCDM in average classification accuracy in one dataset. It is believed that under the circumstance that the real distribution of the data sets cooperate with the supposed distribution of the bayes classifier, it will give the minimum classification error. So, in these cases, this classifier might get some benefit from the distribution of the datasets. It is also worth to point out that for the credit-a dataset, the performances of Naïve Bayes are far worse than those of all other classifiers.

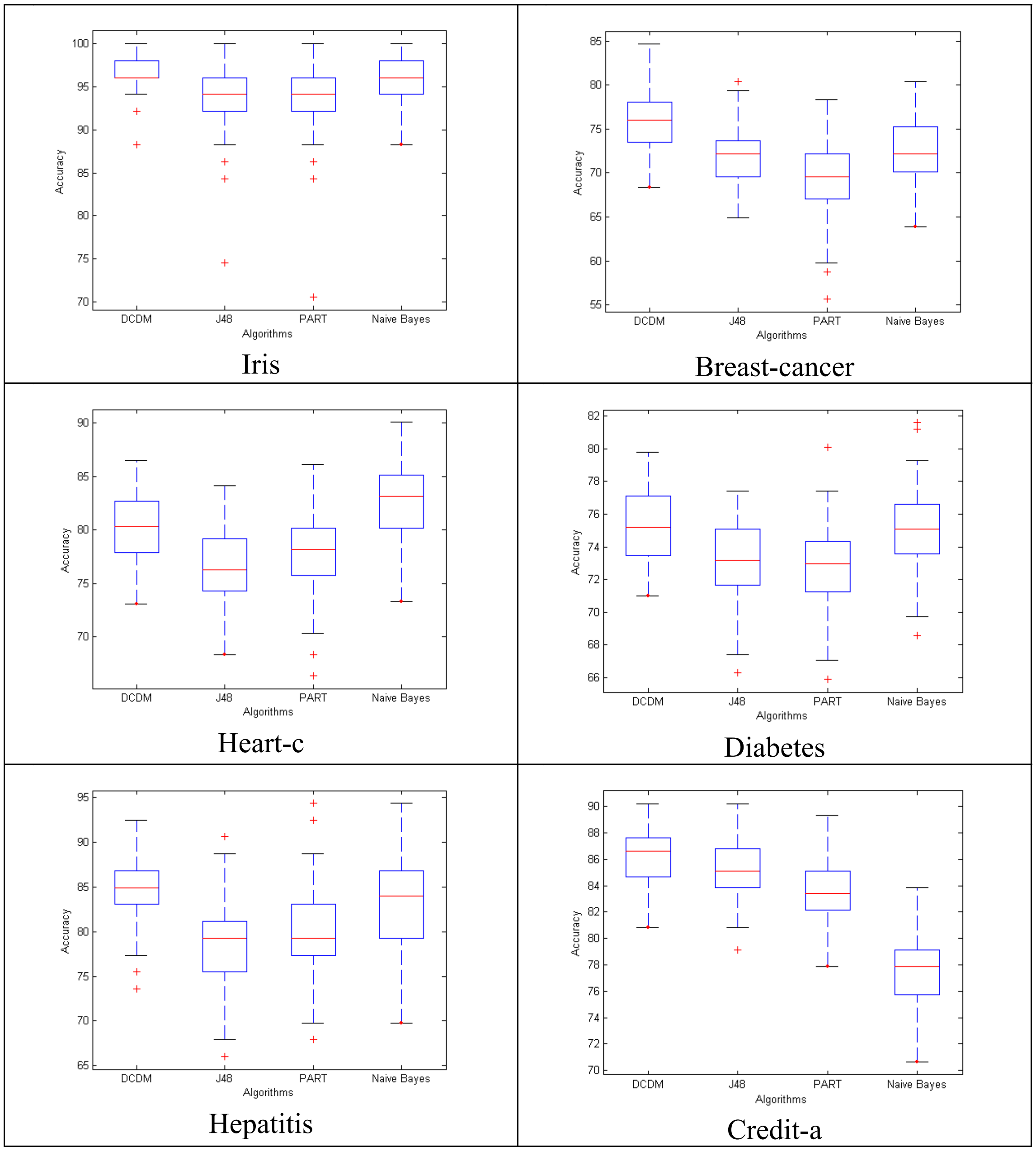


Figure 4.8: Box plots

T-Test: Apart from the box plots, to justify the performance of DCDM statistically, a paired t-test (Montgomery *et al*, 2001) has been performed and the results are given in Table 4.14. For all of the paired t-tests, DCDM was used as the control group and other classifiers as the treatment groups. The alpha level has been chosen as 0.05. Similar to the results shown in the box plots, by observing the P-values obtained from the t-tests, conclusion can be drawn that the classification performance of DCDM is better than those of the other two rule-based classifiers on each of the dataset. For Naïve Bayes, the classification performance of DCDM is significantly worse than those of it on heart-c; however, on the datasets of breast-cancer, and credit-a, it is outperformed by DCDM greatly.

Table 4.14 The P-values of all classifiers on the datasets

Dataset	C 4.5	PART	Naïve Bayes
Iris	1.22×10^{-12}	8.17×10^{-15}	1.48×10^{-6}
Breast-Cancer	1.02×10^{-22}	2.55×10^{-26}	1.21×10^{-24}
Heart-c	3.74×10^{-12}	2.11×10^{-5}	(-) 1.02×10^{-8}*
Diabetes	1.47×10^{-12}	4.05×10^{-15}	0.15
Hepatitis	5.63×10^{-20}	2.20×10^{-14}	0.05
Credit-a	1.17×10^{-8}	2.73×10^{-21}	5.08×10^{-51}

* The minus denotes that the sample mean of results from DCDM is less than that of the classifier under comparison.

4.9.2 Comparisons with other rule-based classifiers

To study the performance of DCDM more thoroughly, the best and the latest results achieved by the rule-based classifiers in literature (including traditional and evolutionary approaches) according to our best knowledge are also included in the comparisons. Although such comparisons are not meant to be exhaustive, it provides a good basis to assess the reliability and robustness of DCDM. Due to the fact that these classifiers use different datasets to evaluate their performances respectively, comparisons are categorized by datasets. All these results are summarized in Table 4.15.

Iris: The co-evolutionary system (GP-Co) proposed by Mendes *et al*., (2001) aimed to discover fuzzy classification rules, in which a GP evolving population and an EA evolving population are co-evolved to generate well-adapted fuzzy rule sets and membership function definitions. Ten-fold cross-validation was used to test this system on the iris data set. The GGP, proposed by Wong (2001), is a flexible knowledge discovery system that applied genetic programming (GP) and logic grammars to learn knowledge in various knowledge representation formalisms. The GBML was proposed by Ishibuchi *et al*., (2001), which is a fuzzy genetic-based machine-learning algorithm that hybrids the Michigan and Pittsburgh approaches. The GBML was tested on several data sets, but only the training accuracy was provided for the iris data. The GPCE, proposed by Kishore *et al*., (2000), is a GP-based technique dedicated to solve multi-category pattern recognition problems. In this algorithm, the *n*-class problem was modeled as *n* two-class problems and GPCE was trained to recognize samples belonging to its own class and reject samples belonging to other classes. The 50 to 50 split percent method was adopted as the validation scheme, and the average results on the validation set over several simulation runs are shown in Table 4.15.

Breast cancer: Fidelis *et al*, (2000) proposed a classification algorithm based on genetic algorithm that discovers comprehensible rules. A fixed encoding scheme is applied to the chromosomes and the mutation operator of GA is under specific design. The breast-cancer dataset is utilized to test the performance of the algorithm by two thirds acting as training set and one third as testing set. GPc, proposed by Tan *et al* (2002b), extends the tree representation of GP to evolve multiple comprehensible classification rules. By utilizing a covering algorithm that employs an artificial immune system-like memory vector, this algorithm produces useful rules and at the same time removes the redundant ones. The validation on the breast-cancer data set used the same partition as Fidelis's work.

The Heart-c dataset: Setiono and Liu (1997) proposed the algorithm of NeuralLinear, which is a system for extracting oblique decision rules from neural networks that have been trained for classification of patterns. The algorithm has been tested on the Heart Disease dataset through ten repetitions of ten-fold cross validation.

The Diabetes dataset: The classification results for Diabetes dataset from several rule-based (Itrule and CN2) and tree-based (CART, AC^2 and Cal5) algorithms (Michie *et al*, 1994) are listed in Table 4.15 for comparisons. Note that these results were obtained by 12-fold cross validation. The GGP (Wong, 2001) was also applied to this dataset and the results are given in Table 4.15. As can be seen from the P-values, DCDM achieves a higher accuracy than GGP generally. However, as GGP is a flexible algorithm, significant performance improvement may be achieved if experts in the relevant domain can incorporate the non-trivial hidden knowledge into its predefined grammars.

Hepatitis: Wang *et al.*, (2000) proposed an evolutionary rule-learning algorithm, called GA-based Fuzzy Knowledge Integration Framework (GA-based FKIF), which utilized genetic algorithms to generate an optimal or near optimal set of fuzzy rules and membership functions from the initial knowledge population. Since the average performance of GA-based FKIF was not provided, only the best result of this algorithm is compared with DCDM.

Credit-a: The classification results for credit-a data set from Itrule, CN2, CART, AC^2 and NeuralLinear are all listed in table X for comparisons.

As shown in Table 4.15, through the extensive comparisons with the best and the latest results achieved by the rule-based classifiers in literature, one can tell that the performance of DCDM is fairly competitive. For the breast-cancer dataset, DCDM outperforms GPc in both classification accuracy and the length of the rule sets. Though Fidelis's algorithm presents rule sets with less number of rules, its classification accuracy falls far below (nearly 20%) of that from DCDM. For the datasets of hepatitis, the results of DCDM are the best no matter in terms of classification accuracy or number of rules in the rule sets. The average classification accuracy on the credit-a of DCDM is slightly worse than that of Itrule and NeuralLinear, but DCDM generates more succinct rule sets than NeuralLinear. For the iris dataset, result from DCDM is also the best among all the classifiers. Discussions about the comparison results will be given in the next section.

Table 4.15 Comparison results

Dataset	Classifier	Accuracy (%)		Number of rules
		Best	Average	
Iris	**DCDM**	**100**	**96.73**	**4**
	GP-Co	-	95.3	-
	GGP	100	91.04	4
	GPCE	96	-	-
	GBML	98.5	-	**-**
Breast cancer	**DCDM**	**84.69**	**76.16**	**5**
	Fidelis *et al*., (2000)	67	-	2
	GPc	73.47	70.65	**12**
Heart-c	**DCDM**	**86.54**	**80.01**	**8**
	NeuralLinear		78.15	6
Diabetes	**DCDM**	**79.77**	**75.31**	**6**
	Itrule	-	75.5	-
	CN2	-	71.1	-
	CART	-	74.5	-
	AC^2	-	72.4	-
	Cal5	-	75.0	-
	GGP	77.95	72.60	14
Hepatitis	**DCDM**	**92.45**	**84.37**	**5**
	GA-based FKIF	92.9	-	-
Credit-a	**DCDM**	**90.21**	**86.28**	**4**
	Itrule	-	86.3	-
	CN2	-	79.6	-
	CART	-	85.5	-
	AC^2	-	81.9	-
	NeuralLinear	-	86.9	6.6

4.10 Discussion and Summary

The proposed DCDM has been examined on 6 datasets obtained from UCI machine learning repository and has produced very good classification results as compared to many existing classifiers. Most of the comparisons were performed statistically using measures such as t-tests and box plots to show the robustness of the proposed classifier. Extensive simulation results show that DCDM has outperformed another two rule-based classifiers (C4.5 rule and PART) on all the testing problems and is very competitive as compared to statistical based techniques, such as Naïve Bayes, in terms of classification accuracy. Furthermore, the box plots results show that DCDM has relatively lesser number of outliers, which indicates that DCDM is relatively more robust and less affected by the random partition of learning and testing sets. It can also be observed from the experiment results that the average number of rules of the rule set produced by DCDM is relatively small as compared to other algorithms. This is an important advantage of DCDM since the comprehensibility of the classification results is directly reflected by its number of rules generally. The performance comparisons to other evolutionary based classifiers (GP-Co, GGP, GBML, GPCE and GA-based FKIF) are mainly restricted by the availability of data, e.g., not all the datasets used in our experiments were tested in other publications. Since there are so many classifiers proposed in literature over the years, it is very difficult, if not impossible, to include every one of them in the comparisons. Therefore the comparisons are not meant to be exhaustive, but to assess the reliability and robustness of DCDM by comparing it with some established methods widely used in literature.

Just as many other rule induction algorithms (Michalski, 1983; Michalski, et al, 1986; Clark & Niblett, 1989; Rivest, 1987), DCDM employs the “separate and conquer” scheme to induction. One shortcoming of this strategy is that it causes a dwindling number of examples to be available as induction progresses, both within each rule and for successive rules. Also, the fact that only single-attribute tests used in rules means that all decision boundaries are parallel to the coordinate axes. With a limited number of examples available, the error of approximation to non-axis-parallel boundaries will be very large. What’s more, taking the form of the decision lists, each rule body in DCDM is implicitly conjoined with the negations of all those that precede it (Domingos, 1996). All of these factors will impair the performance of DCDM on the problems with large number of classes. Apart from this, by incorporating the distributed technology, the efficiency of the coevolutionary algorithm has been significantly enhanced in DCDM, however, to make the search thoroughly, additional computational time is still required as faced by most evolutionary algorithms.

4.11 Conclusion

This chapter has proposed a distributed coevolutionary data mining (DCDM) system for rule discovery. On a distributed platform, the rule population and several rule set populations coevolve in a cooperative manner. By incorporating the coevolutionary algorithm with the distributed technology, not only good classification results can be achieved, but also the efficiency of the evolutionary algorithms can be greatly enhanced. The proposed DCDM has been extensively validated upon 6 datasets obtained from UCI

Machine Learning Repository, and the results have been analyzed both qualitatively and statistically. Comparison results show that the proposed DCDM produces comprehensible and good classification rules for all the datasets, which are very competitive or better than many classifiers widely used in literature.

Chapter 5

Conclusions and Future Works

5.1 Conclusions

In this thesis, two rule-based classification algorithms are presented in which the first one is a two-phased evolutionary approach and the second is a distributed co-evolutionary classifier. The classification performances and the efficiency of the evolution process are the two major considerations of the both algorithms.

In the two-phased approach, a hybrid evolutionary algorithm is utilized in the first phase to confine the search space by evolving a pool of good candidate rules, e.g., genetic programming is applied to evolve nominal attributes for free structured rules and genetic algorithm is used to optimize the numeric attributes for concise classification rules without the need of discretization. These candidate rules are then used in the second phase to optimize the order and number of rules in the evolution for forming accurate and comprehensible rule sets. Good simulation results on three medical datasets show that the algorithms can be used as an assistant tool in clinical practice for better understanding and prevention of unwanted medical events.

In the co-evolutionary system, by utilizing the existing Internet and hardware resources, distributed computing is naturally incorporated into the coevolutionary algorithm to enhance its concurrent processing and performance. Through the inter-communications between the different species (rules and rule sets), the cooperation is conducted in a more effective and efficient way. Rules thus generated are all crucial to the problem, which makes it easy to find the resultant rule set with a fairly good performance. The proposed distributed coevolutionary classifier is extensively validated upon 6 datasets obtained from UCI machine learning repository, which are representative artificial and real-world data from various domains. Comparison results show that the algorithm produces comprehensible and good classification rules for all the datasets, which are very competitive or better than many classifiers widely used in literature.

5.2 Future works

Based on the work in this thesis, there are some possibilities for future research and investigation. On-going work can include the development of peer-to-peer (p2p) computing using JXTA (Juxtapose) technology to improve the performance of the both algorithms. The use of advanced application server such as BEA Weblogic could also enhance the performance and scalability, and features of the server such as cluster and integrated Java message service could be explored to further enhance the efficiency of the evolutionary computation.

References

Andre, D., and Koza, J. R., "Parallel Genetic Programming on a Network of Transputers", *Workshop on Genetic Programming: From Theory to RealWorld Applications*, University of Rochester, National Resource Laboratory for the Study of Brain and Behavior, Technical Report, vol. 95-2, pp.111-120, 1995.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D., *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and its Applications*. San Francisco, CA: Morgan Kaufmann, 1998.

Bojarczuk, C. C, Lopes, H. S., and Freitas, A. A. 'Genetic programming for knowledge discovery in chest-pain diagnosis', *IEEE Engineering in Medicine and Biology Magazine*, vol. 4, no. 19, pp. 38-44, 2000.

Brameier, M., and Banzhaf, W., 'A comparison of linear genetic programming neural networks in medical data mining', *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 17-26, 2001.

Cattral, R., Oppacher, F. and Deugo, D., 'Rule acquisition with a genetic algorithm', *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 125-129, 1999.

Cantú-Paz, E., 'A survey of parallel Genetic Algorithms', *Calculateurs paralleles, reseaux et systems repartis,* Paris: Hermes, vol. 10, no. 2, pp. 141-171, 1998.

Cestnik, G., Konenenko, I., Bratko, I., 'Assistant-86: a knowledge-elicitation tool for sophisticated users' in Bratko, Lavrac (Eds.), *Marchine Learning*, Wilmslow: Sigma Press, pp. 31-45, 1987.

Chambers, J. M., Cleveland, W. S., Kleiner, B., and Turkey, P. A. *Graphical Methods for Data Analysis*, Wadsworth & Brooks/Cole, Pacific CA, 1983.

Chen, Y. W., Nakao, Z., and Xue F., 'A parallel genetic algorithm based on the island model for image restoration', *The 13th International Conference on Pattern Recognition*, vol. 3, pp. 694-698, 1996.

Chong, F. S., *A Java based distributed genetic programming on the Internet*, Master Thesis, School of Computer Science, University of Birmingham, 1997.

Clark, P., and Niblett, T., 'The CN2 induction algorithm', *Machine Learning*, vol. 3, pp. 261-283, 1989.

Congdon, C. B. 'Classification of epidemiological data: a comparison of genetic algorithm and decision tree approaches', *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, pp. 442-449, 2000.

Cristea, V., and Godza, G., 'Genetic algorithms and intrinsic parallel characteristics', *IEEE Congress on Evolutionary Computation,* vol. 1, pp. 431-436, 2000.

De Falco, I., Della Cioppa, A., and Tarantino, E., 'Discovering interesting classification rules with genetic programming', *Applied Soft Computing*, vol. 23, pp. 1-13, 2002.

De Jong K. A., Spears, W. M. and Gordon, D. F., 'Using genetic algorithms for concept learning', *Machine Learning*, vol. 13, pp. 161-188, 1993.

Domingos, P., 'Unifying instance-based and rule-based induction', *Machine Learning*, vol. 24, pp. 141-168, 1996.

Duda, Richard O., Hart, Peter E. and Stork, David G.. *Pattern Classification*, John Wiley and Sons, 2nd edition, 2001

Fayyad, U., “Data Mining and Knowledge Discovery in Databases: Implications for Scientific Databases”, In *Proceedings of the ninth International Conference on Scientific and Statistical Database Management*, pp. 2-11, 1997

Fidelis, M. V., Lopes, H. S., and Freitas, A. ‘Discovering comprehensible classification rules with a genetic algorithm’, *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 805-810, 2000.

Frank, E. and Witten, I. H., ‘Generating accurate rule sets without global optimization’, *Proceedings of the Fifteenth International Conference Machine Learning (ICML'98),* pp. 144-151, 1998.

Freitas, A. A., ‘A survey of evolutionary algorithms for data mining and knowledge discovery’, A. Ghosh and S. Tsutsui. (Eds.), *Advances in Evolutionary Computation*. Springer-Verlag, 2002.

Garner, S. R. ‘WEKA: The Waikato environment for knowledge analysis’, *Proceeding of the New Zealand Computer Science Research Students Conference*, pp. 57-64, 1995.

Giordana, A. and Neri, F., ‘Search-intensive concept induction’, *Evolutionary Computation,* vol. 3, no. 4, pp. 375-416, 1995.

Goldberg, D. E., “Sizing populations for serial and parallel genetic algorithms”, In Schaffer, J. D. (Editor), *The Third International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann Publishers Inc., pp. 70-79, 1989.

Howard, L. M., and D’Angelo, D. J. ‘The GA-P: a genetic algorithm and genetic programming hybrid’, *IEEE Expert,* pp. 11-15, June 1995.

Hruschka, E. R., and Ebecken, N. F. F., ‘A clustering genetic algorithm for extracting rules from supervised neural network models in data mining tasks’, *International Journal of Computers, Systems and Signals*, vol. 1, issue 1, pp. 17-29, 2000.

Hu, Y. J. 'Constructive induction covering attributes spectrum', In H. Liu and H. Motoda (Eds.), *Feature Extraction Construction and Selection: A Data Mining Perspective*, Norwell, MA: Kluwer Academic Publishers, pp. 257-269, 1998.

Ishibuchi, H., Nakashima T. and Murata, T., 'Three-objective genetic-based machine learning for linguistic rule extraction', *Information Sciences*, vol. 136, pp.109-133, 2001.

Janikow, C. Z., 'A knowledge-intensive genetic algorithm for supervised learning', *Machine Learning*, vol. 13, pp. 189-228. 1993.

John, G.H. and Langley, P., 'Estimating continuous distributions in Bayesian classifiers', *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 338-345, Morgan Kaufmann, San Mateo, 1995.

Kim, K. J., and Han, I. 'Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index', *Expert Systems with Applications*, vol. 19, issue 2, pp. 125-132, 2000.

Kishore, J.K., Patnaik, L.M., Mani, V. and Agrawal, V.K., 'Application of genetic programming for multicategory pattern classification', *IEEE Transactions on Evolutionary Computation*, vol. 4, issue 3, pp. 242-258, 2000.

Kohavi, R. 'The power of decision tables', In Lavrae, N. and Wrobel, S. (Eds), *Machine Learning; ECML-95: 8th European Proceedings European Conference on Machine Learning,* Heraclion, Crete, Greece, 1995.

Koza, J. R. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press, 1992.

Levine, D. (1995). *Users Guide to the PGAPack Parallel Genetic Algorithm Library*, ANL-95-18. Available at (http://www.mcs.anl.gov/pgapack.html).

Liu, Y., Yao, X., Zhao, Q.F. and Higuchi, T., 'Scaling up fast evolutionary programming with cooperative coevolution', *IEEE Congress on Evolutionary Computation*, IEEE Press, Piscataway, NJ, USA, pp. 1101-1108, May 2001.

Marmelstein, R. E., Lamont, G. B., GRACCE: A Genetic Environment for Data Mining, *Artificial Neural Networks in Engineering Conference*, 1998.

Meesad, P., and Yen, G. G. 'A hybrid intelligent system for medical diagnosis', *Proceedings of International Joint Conference on Neural Networks*, vol. 4, pp. 2558-2563, 2001.

Mendes, R. R. F., Voznika, F. B., Freitas A. A. and Nievola, J. C., 'Discovering fuzzy classification rules with genetic programming and co-evolution', *Principles of Data Mining and Knowledge Discovery (Proc. 5th European Conf., PKDD 2001) - Lecture Notes in Artificial Intelligence 2168*, pp. 314-325. Springer-Verlag, 2001.

Meta Group Consulting, *CORBA VS DCOM: Solutions for Enterprise*, 1998.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, London: Kluwer Academic Publishers, 1994.

Michalski, R. S., 'A theory and methodology of inductive learning', *Artificial Intelligence*, vol. 20, pp. 111-161, 1983.

Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N., 'The multi-purpose incremental learning system AQ15 and its testing application to three medical domains', *Proceedings of the Fifth International Conference on Artificial Intelligence*, pp. 1041-1045, Philadelphia, PA: AAAI Press, 1986.

Michie, D., Spiegelhalter, D.J., and Taylor, C.C., *Machine Learning, Neural and Statistical Classification*, London: Ellis Horwood, 1994.

Mitchell, T. M., *Machine Learning.* McGraw Hill, 1997.

Montgomery, D. C., Runger G. C., Hubele, N. F., *Engineering Statistics*, New York: Wiley, John & Sons, 2nd Edition, 2001.

Moriarty, D. E. *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*, The University of Texas at Austin, Jan. 1997.

Nang, J. and Matsuo K., 'A survey on the parallel genetic algorithms', *Journal of the Society of Instrument and Control Engineers,* vol. 33, no. 6, pp. 186-191, 1994.

Noda, E., Freitas, A. A., and Lopes, H. S., 'Discovering interesting prediction rules with a genetic algorithm', *IEEE Congress on Evolutionary Computation*, pp. 1322-1329, Washington D.C., USA, July, 1999.

Paechter, B. and Back, T., "A Distributed Resources Evolutionary Algorithm Machine (DREAM)", *IEEE Congress on Evolutionary Computation,* vol. 2, pp. 951-958, 2000.

Paredis, J., 'Coevolutionary computation', *Artificial Life*, vol. 2, pp. 355-375, 1995.

Peña-Reyes, C. A., and Sipper, M. 'A fuzzy-genetic approach to breast cancer diagnosis', *Artificial Intelligence in Medicine*, vol. 17, issue 2, pp. 131-155, 1999.

Peña-Reyes, C. A., and Sipper, M., 'Fuzzy CoCo: A Cooperative-Coevolutionary approach to fuzzy modeling', *IEEE Transactions on Fuzzy System*, vol. 9, no. 5, October 2001.

Polo, A. R. and Hasse, M., 'A genetic classifier tool', *Proceedings of the 20th International Conference of the Chilean Computer Science Society*, pp. 14-23, 2000.

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

Prechelt, L. 'Some notes on neural learning algorithm benchmarking', *NeuralComputing*, vol.9, no.3, pp. 343-347, 1995.

Rivera, W., “Scalable Parallel Genetic Algorithms”, *Artificial Intelligence Review*, vol. 16, pp. 153-168, 2001.

Rivest, R. L., ‘Learning decision lists’, *Machine Learning*, vol. 2, pp. 229-246, 1987.

Rosin, C. D. and Belew, R. K. ‘New methods for competitive coevolution’, *Evolutionary Computation*, vol. 5, no. 1, pp. 1-29, 1997.

Rouwhorst, S. E., and Engelbrecht, A. P. ‘Searching the forest: using decision trees as building blocks for evolutionary search in classification databases’, *IEEE Congress on Evolutionary Computation,* vol. 1, pp. 633-638, 2000.

Setiono, R. and Liu, H. ‘NeuroLinear: From neural networks to oblique decision rules’, *Neurocomputing*, vol. 17, pp. 1-24, 1997.

Setiono, R. ‘Generating concise and accurate classification rules for breast cancer diagnosis’, *Artificial Intelligence in Medicine*, vol. 18, no. 3, pp. 205-219, 2000.

Sleem, A., Ahmed, M., Kumar, A., and Kamel, K., ‘Comparative study of parallel vs distributed genetic algorithm implementation of ATM network environment’, *The Fifth IEEE Symposium on Computers and Communications*, pp. 152-157, 2000.

Street, W. N., Wolberg, W. H., and Mangasarian, O. L. ‘Nuclear Feature Extraction For Breast Tumor Diagnosis’, *In IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, San Jose, CA, vol. 1905, pp. 861-870, 1993.

Sun Microsystems Inc. *J2EE tutorial*, 2001.

Tanev, I., Uozumi, T., and Ono, K., “Parallel Genetic programming: Component Object-based Distributed collaborative approach”, *The 15th International Conference on Information Networking*, pp. 129-136, 2001.

Tan, K. C, Tay, A., Lee, T. H. and Heng, C. M. ‘Mining multiple comprehensible classification rules using genetic programming’, *IEEE Congress on Evolutionary Computation*, Honolulu, Hawaii, pp. 1302-1307, 2002a.

Tan, K. C., Khor, E. F., Cai, J., Heng, C. M. and Lee, T. H., ‘Automating the drug scheduling of cancer chemotherapy via evolutionary computation’**,** *Artificial Intelligence in Medicine,* vol. 25, issue 2, pp. 169-185, 2002b.

Tan, K. C, Yu, Q., Heng, C. M. and Lee, T. H., ‘Evolutionary computing for knowledge discovery in medical diagnosis’, *Artificial Intelligence in Medicine*, vol. 27, issue 2, pp. 129-154, 2003.

Tomassini, M. and Fernandez, F., “An MPI-based tool for distributed genetic programming”, *IEEE International Conference on Cluster Computing*, pp. 209-216, 2000.

Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, N.Y., 1995.

Wang, C. H., Hong, T. P., and Tseng, S. S., ‘Integrating membership functions and fuzzy rule sets from multiple knowledge sources’, *Fuzzy Sets and Systems*, vol. 112, Issue 1, pp. 141-154, 2000.

Witten, I. H., and Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, CA: Morgan Kaufmann Publishers, 1999.

Wong, M. L., and Leung, K. S., *Data Mining Using Grammar Based Genetic Programming and Applications*, London: Kluwer Academic Publishers, 2000.

Wong, M. L., ‘A flexible knowledge discovery system using genetic programming and logic grammars’, *Decision Support Systems*, vol. 31, pp. 405-428, 2001.

Yao, X. and Liu, Y. ‘A new evolutionary system for evolving artificial neural networks’, *IEEE Transactions on Neural Networks*, vol. 8, no. 3, May 1997.

Yoshida, N. and Yasuoka, T., ‘Multi-GAP: Parallel and distributed genetic algorithms in VLSI’, *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, pp. 571-576, 1999.

List of Publications

K.C. Tan, Q. Yu, C.M. Heng, T. H. Lee. “Evolutionary computing for knowledge discovery in medical diagnosis.” *Artificial Intelligence in Medicine,* vol.27, issue 2, pp.129-154, 2003.

Yu, Q., Tan, K. C. and Lee, T. H., “An evolutionary algorithm for rules discovery in data mining”, *Evolutionary Computing in Data Mining*, A. Ghosh and L. C. Jain (Eds.), Physica-Verlag, Germany, 2004.