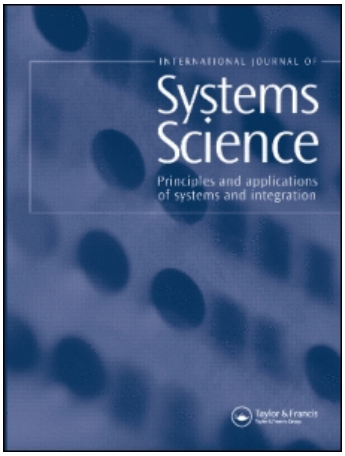


International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713697751

A coevolutionary algorithm for rules discovery in data mining

K. C. Tan [a]; Q. Yu [a];J. H. Ang [a]
[a] Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576

To cite this Article Tan, K. C. , Yu, Q. andAng, J. H.(2006) 'A coevolutionary algorithm for rules discovery in data mining', International Journal of Systems Science, 37: 12, 835 — 864
To link to this Article: DOI: 10.1080/00207720600879641
URL: http://dx.doi.org/10.1080/00207720600879641

International Journal of Systems Science
Vol. 37, No. 12, 10 October 2006, 835–864

Taylor & Francis
Taylor & Francis Group

A coevolutionary algorithm for rules discovery in data mining

K. C. TAN*, Q. YU and J. H. ANG

Department of Electrical and Computer Engineering, National University of Singapore,
4 Engineering Drive 3, Singapore 117576

(*Received 1 August 2003; in final form 8 June 2006*)

One of the major challenges in data mining is the extraction of comprehensible knowledge from recorded data. In this paper, a coevolutionary-based classification technique, namely COevolutionary Rule Extractor (CORE), is proposed to discover classification rules in data mining. Unlike existing approaches where candidate rules and rule sets are evolved at different stages in the classification process, the proposed CORE coevolves rules and rule sets concurrently in two cooperative populations to confine the search space and to produce good rule sets that are comprehensive. The proposed coevolutionary classification technique is extensively validated upon seven datasets obtained from the University of California, Irvine (UCI) machine learning repository, which are representative artificial and real-world data from various domains. Comparison results show that the proposed CORE produces comprehensive and good classification rules for most datasets, which are competitive as compared with existing classifiers in literature. Simulation results obtained from box plots also unveil that CORE is relatively robust and invariant to random partition of datasets.

Keywords: Evolutionary algorithms; Data mining; Classification

1. Introduction

With the rapid advancement in storage device technology, data accumulate at a speed unmatchable by the processing capability of humans. In order to make these raw data useful, improved learning techniques become indispensable. This has led to the emergence of a field named data mining. Data mining is an automated process of extracting structured knowledge from databases, which is often referred to as an essential step in the overall process of discovering useful knowledge from data, called knowledge discovery from database (KDD) (Fayyad 1997, Liu and Motoda 1998). In recent years, there have been numerous attempts to apply evolutionary computation techniques in data mining to tackle the problem of knowledge extraction and classification (Hruschka and Ebecken 2000, Wong and Leung 2000, Tan *et al.* 2002a) or to accomplish tasks in different domains (Banzhaf *et al.* 1998, Cattral *et al.* 1999, Pozo and Hasse 2000, Brameier and Banzhaf 2001). Unlike traditional gradient-guided data mining techniques, evolutionary computation techniques intelligently search the solution space by evaluating performances of multiple candidate solutions simultaneously and approach the global optimum in a non-deterministic manner. Although Evolutionary Computation (EC) techniques play an important role in several areas of data mining domain, they have achieved more popularity for rule based classification (rule induction), for 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 postcondition (Mitchell 1997). Apart from that, these techniques are also able to handle attributes interactions much better than most greedy rule induction algorithms (Freitas 2001, 2002b). Wong and Leung (2000) proposed a grammar-based Genetic Programming (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,

*Corresponding author. Email: eletankc@nus.edu.sg

International Journal of Systems Science
ISSN 0020–7721 print/ISSN 1464–5319 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207720600879641

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. Carvalho and Freitas (2002, 2004) use a genetic algorithm specially designed for discovering small-disjunct rules to cope with the error prone problem often encountered in small disjunct rules used for classificiation in data mining tasks. This is motivated by the findings that interactions are usually the main reason behind the problems of the small disjunct rules. Other Evolutionary Algorithm (EA) approaches for generating classification rules in data mining include Congdon (2000) and Fidelis *et al*. (2000).

There are two ways of encoding rules, i.e. Michigan and Pittsburgh approaches (Michalewicz 1994). The Michigan approach represents a rule while the Pittsburgh approach represents a rule set formed from a combination of rules. Some algorithms that use the Michigan approach are found in GGP (Wong and Leung 2000, De Falco *et al*. 2002), XCS (Wilson, 1995) and XCSR (Wilson, 2000), while those that use Pittsburgh approach are Genetic Algorithm Based Concept Learner (GABIL) (De Jong *et al*. 1993) and Building block approach to Genetic Programming (BGP) (Rouwhorst and Engelbrecht 2000). In XCS, the classifier fitness is based on the accuracy of its prediction and is incorporated with niche GA. The application of XCS in the data mining field can be seen in Saxon and Barry (2000) and Barry *et al*. (2004), and its performance in terms of noise rejection and generalized conditions can be seen in Lanzi and Colombetti (1999) and Lanzi and Perrucci (1999). In a modified version of XCS, XCSR is able to take in real value inputs.

If rule interaction is the main objective, the Pittsburgh approach would be a better choice, but of the motivation is to find a small number of rules with high fitness that is evaluated independently of each other, the Michigan approach is more suitable (Noda *et al*. 1999, Freitas 2002a). As the Pittsburgh approach encodes the whole rule set, the chromosome representing it will be much longer than the chromosome representing the Michigan approach. This enlarges the search space and time taken for the Pittsburgh approach to find a good solution. Also, special care needs to be taken when applying the variation operators as the chromosomes are more complex than in Michigan approach. However, one of the shortcomings of the Michigan approach is that there is no consideration for rules interactions (Freitas 2002a). Niching methods such as token competition is also often needed while applying the Michigan approach in order to maintain the diversity of population (Wong and Leung 2000).

To utilize advantages of both approaches and to minimize the drawbacks of each, the two approaches can be applied together in a certain way. Ishibuchi *et al*. (2001) proposed a multi-criteria genetic algorithm for extraction of linguistic fuzzy rules that considers both the accuracy and length of a rule set. There is also an extended version of this work, where local search and rule weight learning are incorporated into the multi-objective genetic algorithm for candidate rule selection (Ishibuchi and Yamamoto 2004). In Ishibuchi *et al*. (2001), a pre-screening technique was employed to generate the candidate rules that are encoded with the Michigan approach. These candidate rules are selected solely based upon the length of fuzzy rules without considering the applicability or usefulness of the rules. These candidate rules are then used to construct rule sets that are encoded with the Pittsburgh approach. Since the rules and rule sets are searched at different stages, it does not necessarily guarantee the cohesiveness (rules when used together are able to achieve higher efficiency than when used individually) of the rules obtained.

One approach to ensure cohesiveness of the solutions is to evolve both elements simultaneously through coevolutionary-based algorithms, which could evolve multiple populations concurrently in data classification (Mendes *et al*. 2001, Peña-Reyes and Sipper 2001). It has been shown that by coevolving a population of fuzzy membership function with a population of GA individuals (Peña-Reyes and Sipper 2001) or GP tree individuals (Mendes *et al*. 2001), better results could be produced compared to those without the coevolution. Unlike existing approaches, a coevolutionary-based rule extraction and classification system, namely COevolutionary Rule Extractor (CORE), is proposed in this paper to coevolve different types of species, e.g. individuals of rules and rule sets in the evolutionary process. It is shown that instead of evolving random rules, the efficiency and performance of the classifier can be improved by coevolving the populations of rules and rule sets. 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 coevolutionary rule extractor is empowered with token competition (Wong and Leung 2000) to generate the pool of candidate rules. With this technique, the number of candidate rules is significantly reduced and the applicability and usefulness of the candidate rules is automatically assured by the niching capability of the token competition. The population of rules is coevolved cooperatively in parallel with a group of co-populations nurturing the rule sets. Because of the difference in targeted solutions, the Michigan and Pittsburgh coding approaches are employed in the main population and co-populations respectively. The performance of the proposed CORE is extensively evaluated upon seven

selected datasets from UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html), which is a widely used benchmark and real-world data repository in data mining and knowledge discovery community. The classification results of the proposed CORE are analysed both qualitatively and statistically, and are compared with many widely used traditional and evolutionary based classifiers.

The rest of the paper is organized as follows. In section 2, the classification task and rules extraction in data mining are introduced. Besides giving a general overview of coevolutionary algorithms, section 3 presents the proposed coevolutionary rule extraction (CORE) algorithm. Various features in CORE such as chromosome structure, co-populations, fitness evaluation and token competition are also described in the section. The problem sets used for validation and the simulation results are presented in section 4. Section 5 presents the discussion and analysis of CORE results. Conclusions are drawn in section 6.

2. Rule extraction and classification in data mining

The extraction of rules from a database not only serves as the basis for rule-based classification but also provides insight and further understanding of the problem in hand with the novel knowledge discovered. From the data mining point of view, rule-based classification is favoured over many conventional classification techniques by the comprehensibility of its classification decision. Given a set of classified examples, the goal of classification is to find a logical description that correctly classifies novel cases. In the classification task, the discovered knowledge is usually represented in the form of decision trees or IF-THEN prediction rules, which have the advantage of being a high-level and symbolic knowledge representation contributing towards the comprehensibility of the discovered knowledge. In this paper, knowledge is represented as multiple IF-THEN rules in a decision rule list. Such rules state that the presence of one or more items (antecedents) implies or predicts the presence of other items (consequents). A typical rule has the form

$$\text{Rule:} \quad \text{If } X_1 \text{ and } X_2 \text{ and } \ldots X_n \text{ then } Y,$$

where X_i, $i \in \{1, 2, \ldots, n\}$ is the antecedent that leads to the prediction of consequent Y. One reason for using classification rules instead of a decision tree is that each rule can be seen as an independent piece of knowledge. New rules can be added to an existing rule set without disturbing those already there. Multiple rules can be combined together to form a set of decision rules. This set of decision rules is usually listed according to the fitness of the rules, with the best rule listed first. When the decision rule list is used to predict a new instance, the best rule will be considered first. If the rule does not match the instance, i.e. the antecedents of the rule do not satisfy the value of the attributes in the instance, then the next rule will be considered. In the case where none of the rules in the decision list satisfies the new instance, the predetermined default prediction will be used. The default prediction class is the largest class in the training set in this case (the class with the largest number of instances). The basic structure of the decision rule list could be built as follows:

$$\begin{aligned}&\text{IF (antecedents)}_1 \text{ THEN class}_1\\&\text{ELSE IF (antecedents)}_2 \text{ THEN class}_2\\&\ldots \text{ELSE class}_{\text{default}}\end{aligned}$$

The discovered decision rules can be evaluated according to several criteria, such as classification accuracy on unlabelled 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 humans to understand and take action on it accordingly. Generally, rules that are incomprehensible to humans are often useless in data mining or knowledge discovery because such rules are not beneficial to the users. Evaluation of a rule depends on its own performance while evaluation of a rule set is the collective performance of the rules that constitutes it.

3. Coevolutionary rule extractor

3.1 *Coevolutionary algorithms*

Coevolutionary algorithms provide an effective way to broaden the application of the traditional evolutionary algorithm (Potter and De Jong 2000, Rosin and Belew 1997). This is especially noticeable when handling large and highly complex problems like coevolutionary algorithms that employ the divide and conquer strategy. Coevolution algorithms can be implemented at different levels, i.e. single-level or two-level coevolutionary, depending on the type of module to be evolved simultaneously (Khare *et al.* 2004). In single-level coevolution (Potter and De Jong 1994, 2000), each evolving subpopulation represents a subcomponent of the problem to be solved and in two-level coevolution, system and modules are simultaneously optimized in separate subpopulations (Moriarty 1997, Khare *et al.* 2004).

There are basically two types of coevolution strategy, i.e. competitive and cooperative. In competitive

coevolution, individuals are made more competitive through evolution, and in cooperative coevolution the aim is to find the individual from which better systems can be constructed. Competitive coevolution often leads to an "arm race", causing the populations to continuously force one another to improve their fitness level (Angeline and Pollack 1993, Rosin and Belew 1997). The competitive coevolution model is analogous to the predator–prey interaction, where the preys model the candidate optimization solutions while the predators model the "fitness cases" of individuals. There is a direct competition between the fitness of an individual of a species to the fitness of another individual of another species, thus an increase in the fitness level of one implies the reduction of another. Hence new strategies are often evolved within the population to sustain its continued existence.

In cooperative coevolution, a large and complex system is divided into many smaller modules to be evolved separately (Potter and De Jong 2000), and these separately evolving species form the basis of the solution to the complex system. The fitness of an individual depends on its ability to collaborate with individuals from other species. Thus, evolutionary pressure would encourage cooperative strategy and individuals.

Much work and research on the performance of how a single population algorithm performs compared to multiple population evolution has been done, and it is well-known that a single population does not perform better for the following reasons: search space is complex and contains many local optima; a problem or solution can be break down; the genotype is to encode fields of different type e.g. nominal, integer, binary, etc.; and components forming the solution demonstrated interdependencies and their presentation are important. These result in the single population algorithm getting stuck in local optima, increase in computational complexity and thus poor explorative performance compared to its counterpart, i.e. coevolution algorithms that are able to handle these issues (Peña-Reyes and Sipper 2001). When different types of fields exist, coevolution can be applied to encode the different types of fields with each field corresponding to one species. The different species then interact to provide the optimum solution. Apart from the computational complexity, there can be expected savings in terms of computational time, as each individual population that exists in the coevolutionary algorithm is significantly much smaller than the single population (Potter and Dejong 2000).

The fundamental concept of evolving a basic species in parallel with more complex species formed from the basic species are employed in CORE, and this is similar to what is being done in Moriarty (1997). In Moriarty (1997), neural networks are evolved using two cooperative coevolutionary populations. One population encodes a single hidden neuron together with weight values to input and output layers and the other population encodes a set of hidden neurons which is made up of individuals from the other population. These two populations co-evolve to form the required neural networks. In CORE, the species of fundamental elements is the population of rules and the species of the complex elements is the population of rule sets. These two coevolving populations are coupled cooperatively by their fitness as the fitness of rule sets greatly depends on the fitness of rules forming the rule sets. By doing so, the rules forms are more relevant and useful for the rules sets, thus rule sets with good classification accuracy can be expected. Details of the proposed CORE are presented in the following sub-sections.

3.2 *Overall structure of CORE*

The pseudocode of CORE is presented in figure 1 to give a complete overview of the coevolutionary algorithm. As can be seen, the learning process consists of two groups of populations that evolve rules and rule sets respectively and cooperatively. The evolution process of these populations is described in figure 1.

The algorithm first builds from the training dataset a gene map (range of values that each attribute is allowed to take) to maintain a mapping of genes to the corresponding attributes in the dataset. The main population will then be initialized according to the gene map to ensure only valid chromosomes are created. Chromosomes in the main population are encoded using the Michigan approach where each chromosome represents a single rule. These chromosomes are variable

```
Build a GeneMap from training dataset
Initialize the main population based on GeneMap
Evaluate each chromosome's fitness
Repeat
    Use tournament selection to select parents from the main
      population for mating pool
    Apply crossover operator base on probability of crossover to
      create offspring
    Apply mutation operator base on probability of mutation to the
      offspring
    Evaluate the fitness of the chromosomes
    Apply token competition to the offspring
    Apply regenerator operator base on probability of regeneration
    Create co-population by selecting chromosomes from the pool
      of candidates and main population
Until maximum number of allowed generation is reached
```

Figure 1. The pseudocode of the coevolutionary rule extractor.

in length, and all the initial chromosomes are evaluated against the training dataset for their fitness before starting the iteration looping. The mating pool is first formed by selecting parents from the main 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 (described in section 3.6) that works as a covering algorithm. The token competition effectively maintains a pool of good rules, i.e. rules that cover the solution space well. As classification problems generally contain not only one but many useful bits of knowledge, 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 and one-tenth of randomly selected chromosomes from the main population will be used to create the co-populations.

The number of co-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 co-populations. Each co-population maintains a number of rule sets with the same number of rules. All chromosomes in the co-populations are encoded with the Pittsburgh approach where each chromosome represents a rule set. The fitness of these rule sets is greatly affected by the rules used. As an algorithm is still in the training phase, to evaluate the co-chromosomes, the classification accuracy on the training set is used. Here, only the mutation operator is applied to evolve the co-chromosomes in order to avoid reproduction of redundant rule sets. After the new main population and co-populations have been evolved, the coevolution will proceed to the next generation and the process will be repeated until the last generation is reached. 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 co-populations to obtain the final optimal rule set. 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.

3.3 *Population and chromosome structure*

The coevolution strategy used two different population structures, i.e. the main population coevolves with the co-populations as illustrated in figure 2. The main population contains subpopulation of each class. For example, if there are 3 classes, the main population consists of 3 subpopulations, with each subpopulation evolving individual rules for each class. Chromosomes in the main population are encoded with the Michigan approach, that is, one chromosome represents a rule. In a given chromosome, each gene is associated with an attribute of the dataset. The structure of the chromosome and gene are depicted in figure 3. As can be seen, the number of genes used to construct the chromosome is variable. Therefore each chromosome does not need to contain all the attributes or to contain the attributes in order. Since most datasets consist of nominal and numeric attributes, two types of genes are included to handle them respectively.

Nominal attributes assume a finite set of ordinal values while numerical attributes take on a continuous range of values. As they are different in nature, different handling techniques are required. To do that, the gene structure is facilitated with 3 fields: attribute index, relation and value. The attribute index is the index of the attribute that the gene corresponds to. The relation field is used to assign the relationship operator for the attribute with respect to the value. For nominal attribute, only equal ($==$) and not equal ($\neq$) operators are used. On the other hand, for numeric attribute, 6 comparison operators: greater than ($>$), greater than or equal ($\geq$), less than or equal ($\leq$), less than ($<$), in-bound ($><$) and out of bound ($<>$) are used. The in bound and out bound operators enable the chromosome to encode numeric attribute with range, i.e. rules with $a < attribute < b$ (in bound) and $attribute < a$ *and* $attribute > b$ (out bound) are possible. The last field is the value of the corresponding attribute. For nominal attribute, the value is a bit string array that is associated with the index of value for the attribute. For example, if an attribute has three possible values, temperature $=$ {low, medium, high}, then the value of [1 0 1] corresponds to the first attribute value (low) and the third attribute value (high). The numeric attribute on the other hand is a scaled real value ranging from 0 to 1. Thus, if a numeric attribute has the exact value ranged from 50 to 68, the 0 and 1 will represent 50 and 68 respectively. In the case of bound relation, the value is an array containing the lower and upper bounds of the attribute. Since nominal attributes are encoded using binary representation, the maximum and minimum limits take on the values of 0 and 1. For consistency, the numeric attributes take on the values bounded by the limits of 0 and 1. The encoding style used here to represent the attribute value in a chromosome is one of the many ways available. Different encoding methods can thus also be applied.

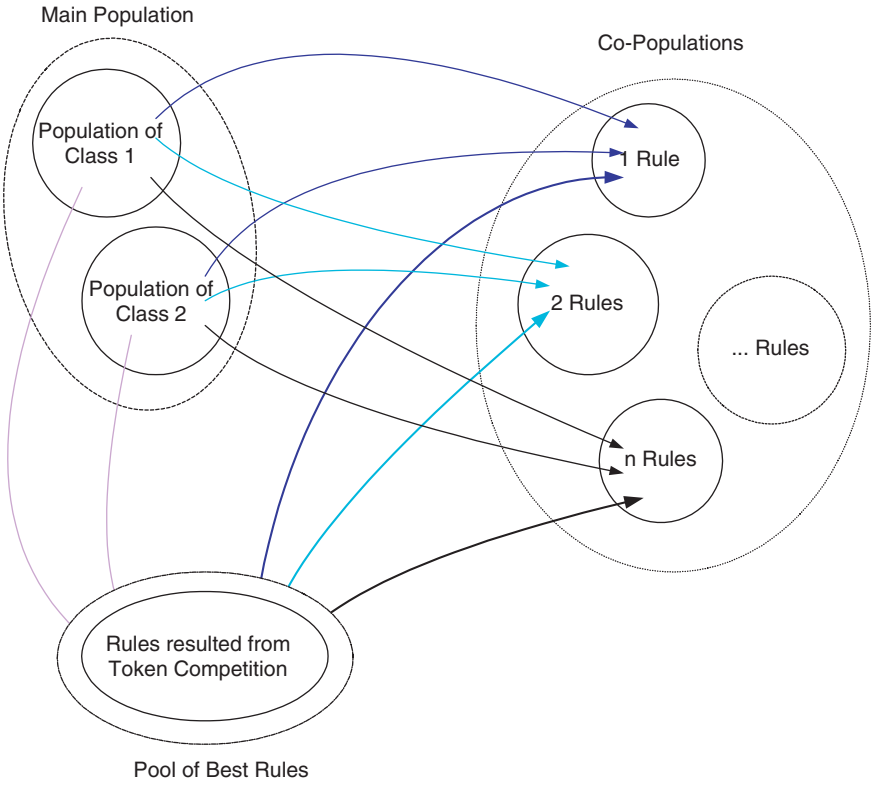


Figure 2. Illustrative structure of populations and coevolution links in CORE.

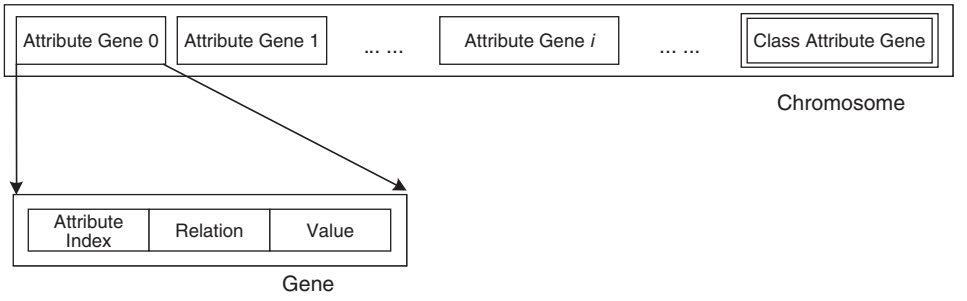


Figure 3. Chromosome structure in CORE.

The CORE algorithm applies a group of co-populations to evolve rule sets with different number of rules. The chromosomes in these co-populations, namely co-chromosomes are encoded with Pittsburgh approach where each co-chromosome is encoded with a rule set. The structure of the co-chromosome is depicted in figure 4. The basic element that builds up the co-chromosomes is the chromosome representing rules from the main population. The number of rules in a co-chromosome depends on which co-population the co-chromosome is attached to. For example, the fourth co-population will only contain co-chromosomes with 4 rules. Note that the default class is also encoded in the co-chromosome although it is not counted as a rule. The default class is not counted as a rule in this case as there are no antecedent elements. The default class is assumed if conditions of previous rules in the rule set are not fulfilled. This makes sure that an instance will definitely be classified by a rule set, thus no instance will be left unclassified. Since both the main and co-populations differed from each other, the genetic operators applied are also different as discussed in the next sub-section.

3.4 *Genetic operators*

The genetic operators applied to the main population are crossover and mutation. Here, the genetic operations

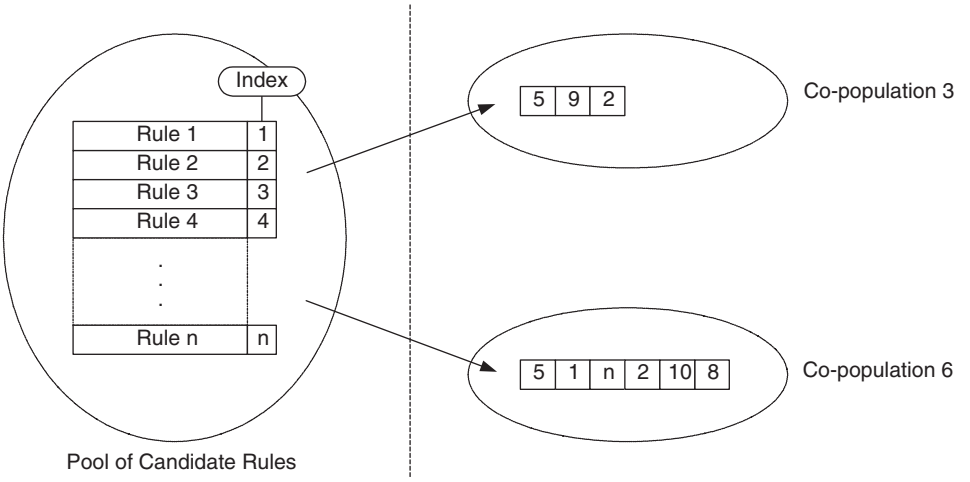


Figure 4. The structure of co-chromosomes in CORE.

take place at two levels, i.e. chromosome level and gene level. At the chromosome level, one point crossover is used for which a random crossover point is selected for each parent chromosome and genes are exchanged. The length of the offspring is not necessarily the same as their parents'. The chromosome-level mutation removes or inserts a newly generated gene into the chromosome. At the gene level, crossover and mutation are different for nominal and numeric attributes. For the nominal gene, the gene-level crossover is a one-point crossover to the bit string array of the parents. On the other hand, the gene level crossover for the numeric gene is a standard real-coded crossover with the following equations,

$$offspring_1 = \alpha parent_1 + (1 - \alpha) parent_2 \quad (1)$$

$$offspring_2 = \alpha parent_2 + (1 - \alpha) parent_1 \quad (2)$$

where α is a random number that can take any value in the range $[0 \ldots 1]$. The gene level mutation is only done on the "value" field and not other fields like "attribute index" and "relation". The mutation operator would make sure that the gene remains valid after mutation. For example, if the "value" field of a gene is a nominal value and can only take "low", "normal" and "high" then mutation occurs such that only these are possible results of the mutation. If the "value" field is a numerical value, a small number is randomly generated and is added or subtracted from it. The mutation operation also makes sure it falls within the bounds of the minimum and maximum value that it can take.

Only the mutation operator is used in evolving the co-population, e.g. one rule is removed from or inserted into the co-chromosome. When the number of rules in a co-chromosome is changed, the co-chromosome will be moved to the correct co-population. For example, if a five-rules co-chromosome in co-population 5 was mutated and one rule was removed from it, then it will be moved to co-population 4.

3.5 *Fitness evaluations*

When a rule or individual is used to classify a given training instance, one of 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 two-class case, with class "Yes" and "No", the four concepts can be easily understood with the following descriptions (Fidelis *et al.* 2000):

- True positive (*tp*): the rule predicts that the class is yes (positive) and the class of the given instance is indeed yes (positive);
- False positive (*fp*): the rule predicts that the class is yes (positive) but the class of the given instance is in fact no (negative);
- True negative (*tn*): the rule predicts that the class is no (negative) and the class of the given instance is indeed no (negative);
- False negative (*fn*): the rule predicts that the class is no (negative) but the class of the given instance is in fact yes (positive).

In the multi-class case, for example, if an instance that belongs to Class 3 is presented to a rule that has Class 5 encoded in its consequent part, and the rule predicts that

the instance does not belong to Class 5 then the concept is a true negative. However, if an instance that belongs to Class 5 is presented to the same rule while it predicts the instance does not belong to Class 5, then the concept is false negative.

Using these concepts, the fitness function used in evaluating the main population of CORE is defined as,

$$\text{Fitness} = \text{penalty} \times \frac{tp}{(tp+fn)} \times \left(1 + \frac{tn}{(tn+fp)}\right) \quad (3a)$$

$$\text{With penalty} = \frac{N}{N+fp}, \quad (3b)$$

where N is the total number of instances in the training set; tp, fp, tn, and fn is true positive, false positive, true negative and false negative, respectively. 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. The penalty factor is included in the fitness function to evaluate the fitness of the combined individuals in the rule set. This is because the Boolean sequential rule list (where rules are considered one after another) is too sensitive and has the tendency of having a large number of false positives (fp) due to the virtual OR connection among the rules. When a rule with large fp is considered first in a rule list, many of the instances will be classified incorrectly. Therefore, the fitness function should be penalized based on the value of fp, e.g. a penalty factor w that tends to minimize fp is included in equation (3). Since the number of rules is fixed for each co-population, there is no need to explicitly formulate the comprehensibility in the fitness evaluation of co-populations. Indeed, the fitness function of the co-populations can be simply formulated as the classification accuracy. In order to evaluate the performance of a rule set, it is necessary to order the rules according to their fitness.

3.6 *Token competition*

Token competition is applied in CORE to evolve different multiple rules for prediction of each class in the dataset as well as to preserve the diversity in the evolution (Wong and Leung 2000). It tries to find rules that are able to cover all instances under a class, if possible, and at the same time to exclude instances not in the class. There is often no single rule that can cover all instances of a class: hence there is a need to discover more rules that can predict all instances of a class, but at the same time do not overlap with instances of another class. The CORE also applied the principle of controlling the fitness of rules via the concept of minimum support as proposed by Tan *et al.* (2002a). Every instance in a dataset is called a token, for which all chromosomes in the population will compete to capture. A chromosome has the chance to capture a token if all its antecedents match that in the token and the class in the token is the class predicted by the chromosome, i.e. a tp concept case. If more than one chromosome is eligible to capture the same token, then only the fittest chromosome will be assigned the token. The adjusted fitness is calculated for each chromosome after the token competition as given by,

$$adjusted\_fitness = fitness \times \frac{actualNumberofCapturedTokens}{numberOfClassOccurences} \quad (4)$$

The term *numberOfClassOccurences* refers to the total number of instances or tokens in the dataset containing the class to be predicted, while *actual NumberofCapturedTokens* refers to the total number of tokens in the dataset that has been captured by the chromosome. Obviously, the relation $actual\ NumberofCapturedTokens \leq numberOfClassOccurences$ is always true.

Table 1. Iris dataset description.

	Min	Mean	Max	Standard dev.	Class correlation
Sepal length	4.3	5.84	7.9	0.83	0.7826
Sepal width	2.0	3.05	4.4	0.43	−0.4194
Petal length	1.0	3.76	6.9	1.76	0.9490
Petal width	0.1	1.2	2.5	0.76	0.9565

4. Case study

4.1 *Experimental setup*

The proposed CORE is validated, based on seven datasets from the UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). These datasets are categorized into three categories (in sections 4.2.1, 4.2.2 and 4.2.3) according to their unique characteristics. The detailed descriptions of each dataset abridged from the documentations in UCI are given in tables 1–7. Table 8 describes the domains of the data together with the classification tasks, while in table 9, the characteristics of each dataset are summarized. Each of these datasets is partitioned into two sets, a training set and a testing set. The training set is used to train CORE, through which its learning capability can be justified. However, a classifier that learns well does not necessarily

Table 2. Hepatitis dataset description.

	Attribute	Description	No. of missing attribute
1	Age	Integer	0
2	Sex	male, female	0
3	Steroid	No, yes	1
4	Antivirals	No, yes	0
5	Fatigue	No, yes	1
6	Malaise	No, yes	1
7	Anorexia	No, yes	1
8	Liver big	No, yes	10
9	Liver firm	No, yes	11
10	Spleen palpable	No, yes	5
11	Spiders	No, yes	5
12	Ascites	No, yes	5
13	Varices	No, yes	5
14	Bilirubin	Continuous	6
15	Alk phosphate	Integer	29
16	Sgot	Integer	4
17	Albumin	Continuous	16
18	Protime	Integer	67
19	Histology	No, yes	0

Table 3. Diabetes dataset description.

	Attribute	Mean	Standard deviation
1	Number of times pregnant	3.8	3.4
2	Plasma glucose concentration	120.9	32.0
3	Diastolic blood pressure (mm Hg)	69.1	19.4
4	Triceps skin fold thickness (mm)	20.5	16.0
5	2-Hour serum insulin (mu U/ml)	79.8	115.2
6	Body mass index	32.0	7.9
7	Diabetes pedigree function	0.5	0.3
8	Age (years)	33.2	11.8

Table 4. Breast cancer dataset description.

	Attribute	Description	No. of missing attribute
1	Age	Integer	–
2	Menopause	lt40, ge40, premeno	–
3	Tumour size	Integer	–
4	Inv-node	Integer	–
5	Node-caps	Yes, no	–
6	Deg-malig	1, 2, 3	8
7	Breast	Left, right	–
8	Breast-quad	Left-up, left-low, right-up, noright-low, central	–
9	Irradiation	Yes, no	1

Table 5. Sick dataset description.

	Attribute	Description
1	Age	Continuous
2	Sex	M, F
3	On thyroxine	f, t
4	Query on thyroxine	f, t
5	On antithyroid medication	f, t
6	Sick	f, t
7	Pregnant	f, t
8	Thyroid surgery	f, t
9	I131 treatment	f, t
10	Query hypothyroid	f, t
11	Query hyperthyroid	f, t
12	Lithium	f, t
13	Goitre	f, t
14	Tumour	f, t
15	Hypopituitary	f, t
16	Psych	f, t
17	TSH measured	f, t
18	TSH	Continuous
19	T3 measured	f, t
20	T3	Continuous
21	TT4 measured	f, t
22	TT4	Continuous
23	T4U measured	f, t
24	T4U	Continuous
25	FTI measured	f, t
26	FTI	Continuous
27	TBG measure	f, t
28	TBG	Continuous
29	Referral source	WEST, STMW, SVHC, SVI, SVHD, other

guarantee it is also good in generalization. In order to evaluate the generalization capability, the rule sets obtained by CORE are applied to the testing set after training. Instances with missing data are discarded. Removing observations with any missing data is one of the ways to handle missing values. This has been widely adopted by many works in literature and one of the reasons for its popularity is due to its easy implementation (Little and Rubin 1987). The CORE was programmed using the Java Developers Kit (JDK 1.3.1) from Sun Microsystems on an Intel Pentium IV 1.4 GHz computer with 128 MB SDRAM (A).

4.1.1 The Fisher's iris dataset. This dataset is perhaps the best-known database to be found in pattern recognition literature. Fisher (1936) is a classic in the field and is referenced frequently to this day. The dataset contains three classes of 50 instances each, where each class refers to a type of Iris flower. One class is linearly separable from the other two; the latter are not linearly

Table 6. Heart disease dataset description.

	Attribute	Description
1	Age (years)	Integer
2	Sex	1 = male, 0 = female
3	Chest pain type	Value 1: typical angina Value 2: atypical angina Value 3: non-anginal pain Value 4: asymptomatic
4	Resting blood pressure (in mm Hg on admission to the hospital)	Integer
5	Serum cholesterol (mg/dl)	Integer
6	Fasting blood sugar > 120 (mg/dl)	1 = true, 0 = false
7	Resting electrocardiographic results	Value 0: normal Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8	Thalach: maximum heart rate achieved	Integer
9	Exercise induced angina	1 = yes, 0 = no
10	Oldpeak = ST depression induced by exercise relative to rest	Continuous
11	Slope of the peak exercise ST segment	Value 1: upsloping Value 2: flat Value 3: downsloping
12	Number of major vessels	Value 0–3: coloured by flourosopy
13	Thal	Value 3: normal Value 6: fixed defect Value 7: reversable defect

Table 7. Australian credit card assessment dataset description.

	Value	No. of missing value
1	a, b (0, 1)	12
2	Continuous	12
3	Continuous	–
4	p, g, gg (1, 2, 3)	6
5	ff, d, i, k, j, aa, m, c, w, e, q, r, cc, x (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)	6
6	ff, dd, j, bb, v, n, o, h, z (1, 2, 3, 4, 5, 6, 7, 8, 9)	9
7	Continuous	9
8	t, f (1, 0)	–
9	t, f (1, 0)	–
10	Continuous	–
11	t, f (1, 0)	–
12	s, g, p (1, 2, 3)	–
13	Continuous	–
14	Continuous	13

separable from each other. Although Fisher's Iris dataset is a rather simple domain compared to the next two categories of databases, it can be regarded as a multi-class (more than two classes) problem since it contains three classes. Through the performance study on this dataset, the classification capability of CORE for multi-class problems could be assessed. The attributes are sepal length, sepal width, petal length and petal width (all in centimeters). The output classes are iris setosa, iris versicolour and iris virginica. There are no missing attribute values in the dataset. Table 1 gives the summary statistics.

4.1.2 The hepatitis dataset. This dataset was collected at Carnegie-Mellon University (Cestnik *et al.* 1987) by Gail Gong and donated to UCI ML repository in 1988. It consists of 155 instances with 32 (20.65%) dead samples and 123 (79.35%) live samples. Each instance contains 19 attributes (table 2), inside which 14 are nominal and 5 are numeric. The hepatitis problem is a complex and noisy dataset as it contains a large number of missing data (there are 167 missing values in total in this dataset). This dataset challenges any rule-based classifier to find the optimal rule set. The learning task is to predict whether a patient with hepatitis will live or die.

4.1.3 The diabetes dataset. This dataset was first collected at Johns Hopkins University by Vincent

Table 8. Classification task descriptions of the datasets.

Dataset	Domain	Classification task
Iris	Botany	Classify 3 species of iris flower based on their physical characteristics
Hepatitis	Medicine	Determine whether a hepatitis patient will live or die according to given medical conditions
Diabetes	Medicine	Determine whether a patient shows sign of diabetes according to World Health Organization Criteria
Breast cancer	Medicine	Determine the patients for whom the cancer will re-occur
Sick	Medicine	Determine whether a patient is sick or not
Heart-c	Medicine	Determine the risk of heart disease given a certain medical condition in patients
Australian credit card assessment	Finance	Determine a certain aspect of credit card applications given other specifications

Table 9. The characteristics of the datasets.

					Attribute characteristics		
Dataset	No. of attributes	No. of classes	No. of instances	% of major class	Numeric	Nominal	Missing
Iris	4	3	150	33	Yes	No	No
Hepatitis	19	2	155	79	Yes	Yes	Yes
Diabetes	8	2	768	65	Yes	No	No
Breast cancer	9	2	286	66	No	Yes	Yes
Sick	29	2	3772	94	Yes	Yes	Yes
Heart-c	13	5	303	55	Yes	Yes	Yes
Australian credit card assessment	14	2	690	56	Yes	Yes	Yes

Sigillito and then constructed by constrained selection from a large database held by the National Institute of Diabetes and Digestive and Kidney Diseases. Among the total 768 instances in this dataset, there are 500 examples of class 1 (tested positive) and 268 of class 2 (tested negative). Eight attributes are included in this dataset, all of which are numeric ones (table 3). The learning task is to predict whether a patient shows signs of diabetes according to World Health Organization criteria.

4.1.4 The breast cancer dataset. This dataset was collected from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia and provided by M. Zwitter and M. Soklic. It includes 201 instances of one class (no recurrence events) and 85 instances of another class (recurrence events). The instances are described by 9 attributes (table 4), which are all nominal. Missing attribute values also exist in this dataset but cover a small portion. The learning task is to predict whether the tumour of a patient will reoccur or not.

4.1.5 The sick dataset. This dataset was supplied by the Garavan Institute and J. Ross Quinlan, New South Wales Institute, Sydney, Australia. Each of the total 3772 instances is represented by 29 attributes, among which 7 are numeric and all the rest are nominal. There are 6064 missing values, which account for 5.4% of the total. The learning task is to predict whether a patient is sick or not. The description for each attribute and its encoding is given in table 5.

4.1.6 The heart disease dataset. This dataset was collected at Cleveland Clinic Foundation and documented by Robert Detrano. Though the database contains 76 attributes, all published experiments refer to using a subset of 14. There are 297 samples in total, after discarding 6 which contain missing values. Each sample in this dataset can be described by 13 attributes (table 6), among which 6 are numeric and 7 are nominal. The learning task is to predict the presence or absence of heart disease given the results of various medical tests carried out on a patient. An integer value of 0 is used to indicate the absence of heart disease while values 1, 2, 3 and 4 show the presence. The distribution of

classes 0, 1, 2, 3 and 4 in the original database is 54%, 18%, 12%, 12% and 4% respectively.

4.1.7 The Australian credit card assessment dataset. This dataset contains 690 examples and was submitted by Quinlan to UCI machine-learning repository. Among the total 14 attributes, 8 are nominal and 6 are numeric. The learning task is to assess applications for credit cards based on these attributes, whether the application should be approved. Due to the confidentiality of the data, all attributes names and values have been changed to meaningless symbols (table 7). Labels have been changed for the categorical attributes, e.g. attribute 4 originally had labels p, g, gg, and these are changed to 1, 2 and 3 in the dataset. This dataset set is interesting as it contains a good mix of attributes. There are continuous and nominal attributes, with a few missing values. The nominal attributes have a small and large number of values. 37 instances (5%) have missing value. These missing values are being replaced by the mode if it is a categorical attribute or else it is replaced by the mean if it is a continuous attribute in the UCI repository.

As indicated by Prechelt (1995), a fuzzy specification of the partitioning of training versus testing data is a big obstacle to reproducing and comparing published machine-learning results. Only indicating the number of examples for each set in the partition is insufficient because the experimental results may vary significantly for different partitions even when the numbers in each set are the same (Yao and Liu 1997). 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 seven datasets, a random seed, which is the same as the number of runs (i.e. the 50th run uses random seed 50), is first used to randomize the order 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. The random number generator used in the experiments is provided with Sun's JDK 1.3.1 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 10 lists the parameter settings in CORE that are applied to all the seven datasets. These parameters have been chosen after some preliminary experiments and heuristics (Tan *et al*. 2003, Tan *et al*. 2006), 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 CORE can perform well over a wide range of datasets. For the algorithm to work better, users can try other values that are suited to the problem at hand. Note that by choosing large values for probability of mutation and probability of regeneration, the algorithm will resemble random search which is undesirable. Any good properties that are passed down from the parents to the offspring during crossover will be lost. There is no guarantee that a good solution can be found within a given period of time.

Table 10. 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 11. Summary of results on the Iris dataset.

	Default setting	The 2nd setting
Training		
Max	100%	100%
Min	90.91%	90.91%
Mean	97.64%	96.97%
StdDev	1.28%	2.17%
Test		
Max	100%	100%
Min	90.20%	90.91%
Mean	96.61%	96.06%
StdDev	2.34%	2.59%
Avg # rules	3.77	3.12
Training Time (s)	151	76

Max: Maximum classification accuracy. Min: Minimum classification accuracy. Mean: Mean classification accuracy.

4.2 *Simulation results and comparisons*

4.2.1 The botany dataset. (A) *Experimental results*. Table 11 summarizes the results of CORE on the Iris data, which include the results obtained by the default parameter settings in table 10 as well as by another set of parameter settings with smaller training time (with a population and generation size of 50). One hundred independent simulation runs have been conducted for both of the settings.

In order to have a clearer view of the classification performance for the default parameter settings over

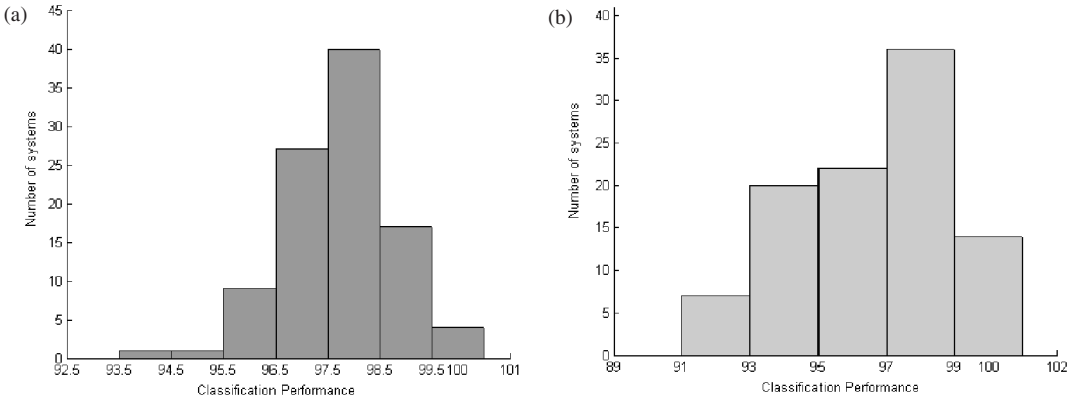


Figure 5. The performance of CORE upon the Iris data: (a) training; (b) testing.

Table 12. The best classification rule set for the Iris data.

	Rule	Fitness	Support factor	Confidence factor
1	IF petallength < 1.9471, THEN class = Iris-setosa	2.0	0.3434	1.0
2	IF petallength >< [1.1, 2.0582], THEN class = Iris-setosa,	2.0	0.3434	1.0
3	IF petalwidth <> [0.1, 1.6747], THEN class = Iris-virginica	1.7031	0.303	0.9375
4	IF petallength >< [4.4422, 4.9938], THEN class = Iris-versicolor	1.4697	0.2424	0.8889
5	IF petallength <> [1.2707, 4.4234], THEN class = Iris-virginica,	1.8084	0.3333	0.8089
6	ELSE class = Iris-versicolor			
Classification accuracy = 100%				

the 100 independent evolutionary runs, the histograms plots for the results are shown in figure 5. Clearly, a normally distributed performance has been achieved by CORE for this dataset. The classification rule set that has the highest predictive accuracy (i.e. the highest classification accuracy on the testing set) on the Iris data is presented in table 12. It should be noted that larger rule sets might not perform better than the smaller rule set on testing set if overfitting has occurred. Due to over training, larger rule sets perform well on the training set however poorly they performed on the test set. Besides the fitness value, the support factor and confidence factor are provided as additional information to show 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. On the other hand, the confidence factor measures the accuracy of the rule. For a rule 'if X then Y' and with a training set of N instances, the support and confidence factors are given as,

$$\text{Support} = \frac{\text{Number of instances with both } X \text{ and } Y}{N} \tag{5}$$

$$\text{Confidence} = \frac{\text{Number of instances with both } X \text{ and } Y}{\text{Number of instances with } X}. \tag{6}$$

It can be seen from the rule set in table 12 that the classification rules are not ordered in any specific pattern since the rule sets were formed randomly during the evolution. As shown in table 12, all the rules in this rule set have relatively high accuracy and confidence factor. It can also be observed that the rule set was constructed of only petal length and petal width attributes, which clearly demonstrates the autonomous attribute selection ability of CORE. Since the rule set is

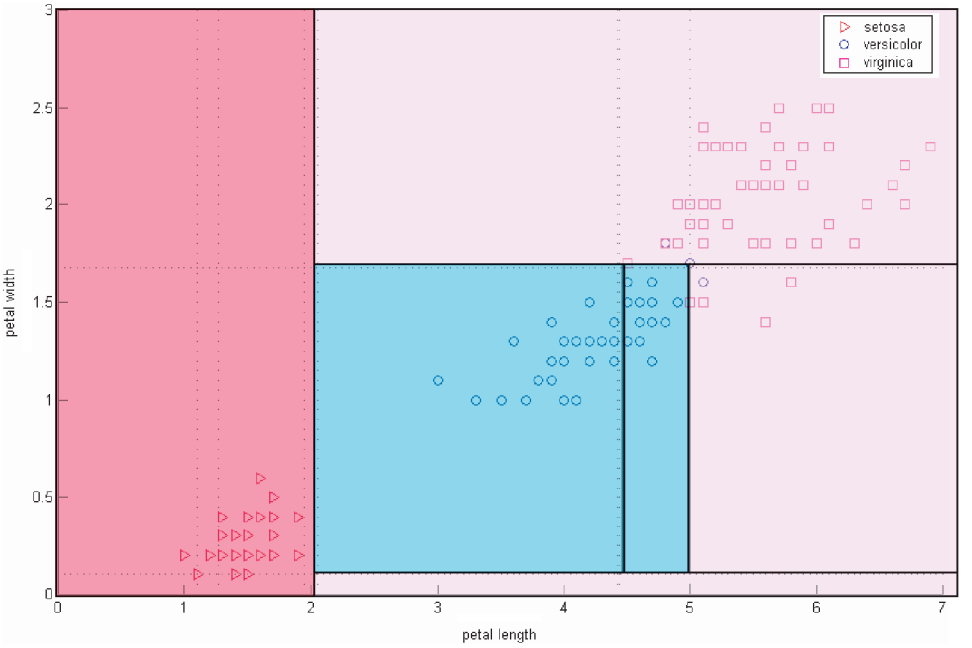


Figure 6. The visualization graph for the rule set of Iris data.

a 2 dimensional function, it can be well visualized as shown in figure 6.

(B) *Comparisons with other works*. As stated in the Introduction, the proposed CORE is capable of generating comprehensible rule sets with good classification accuracy. For comparison, two famous rule-based machine-learning algorithms C4.5 rules, e.g. J48 in WEKA (Witten and Frank 1999) and PART as well as a statistical classifier Naïve Bayes have been applied to the seven datasets. The first two algorithms are chosen due to their rule-based characteristics as offered in CORE. Comparisons between these two algorithms and CORE 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 CORE in terms of classification ability. A brief description of these methods is given below.

- C4.5 system proposed by Quinlan (1993) is a landmark decision tree program, which is one of the machine learning methods most widely used in practice to date.
- PART is a rule-learning scheme, which can generate good classification rules (Frank and Witten 1998).
- Naïve Bayes utilizes Bayesian techniques, which have recently been used by many machine learning researchers (John and Langley 1995).

To study the performance of CORE 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 CORE.

The classification results for all algorithms under comparisons are listed in table 13. The co-evolutionary system (GP-Co) proposed by Mendes *et al.* (2001) aimed to discover fuzzy classification rules. Two evolving populations, based on GP and EA, co-evolve to generate fuzzy rule sets and membership function definitions. These two populations 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 dataset. The GGP was proposed by Wong (2001), which is a flexible knowledge discovery system that applied genetic programming (GP) and logic grammars to learning 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 approach. Ishibuchi *et al.* (2001) tested this algorithm on several datasets, but only the training accuracy was provided for the Iris data. The GPCE was proposed by Kishore *et al.* (2000), which is a GP-based technique dedicated to solving multi-category pattern recognition problems. In this algorithm,

Table 13. Performance comparisons for the Iris dataset.

Algorithm	Reference	# Rules	Avg accuracy	Best accuracy	Standard deviation
CORE	–	3.77	96.61%	100%	2.35%
C4.5	(Quinlan 1993)	3.87	93.67%	100%	3.73%
PART	(Frank and Witten 1998)	4.15	93.94%	100%	3.93%
Naïve Bayes	(John and Langley 1995)	–	95.47%	100%	2.93%
GP-Co	(Mendes *et al.* 2001)	–	95.3%	–	7.1%
GGP	(Wong 2001)	4	94.24%	100%	3.57%
GBML	(Ishibuchi *et al.* 2001)	5	–	98%	–
GPCE	(Kishore *et al.* 2000)	–	96%	–	–
XCS	(Bernadó *et al.* 2001)	–	94.7%	–	5.3%
XCSR	(Bacardit and Garrell 2003)	3.5	95.2%	–	2.3%

the *n*-class problem was modelled 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 50/50 split percent method was adopted by Kishore *et al.* (2000) as the validation scheme, and the average results on the validation set over several simulation runs are shown in table 13. Two well known learning classifiers, XCS and XCSR which is another version of XCS, are also included for comparison. XCS developed by Wilson (1995) is a classifier system based on the Michigan coding approach. The results presented are taken from Bernadó *et al.* (2001). XCSR (Wilson 2000) is a version of XCS which takes in real value inputs. The results presented are taken from Bacardit and Garrell (2003).

As shown in figure 7, the simulation results are represented in box plot format (Chambers *et al.* 1983) to visualize the distribution of simulation data 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 represents 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 outliers. It can be seen that although the best accuracy of 100% has been achieved by all algorithms, CORE is superior to other algorithms in terms of average accuracy by having the smallest variance. Furthermore, CORE is shown to be statistically more stable as it produced no outlier for the 100 simulation runs.

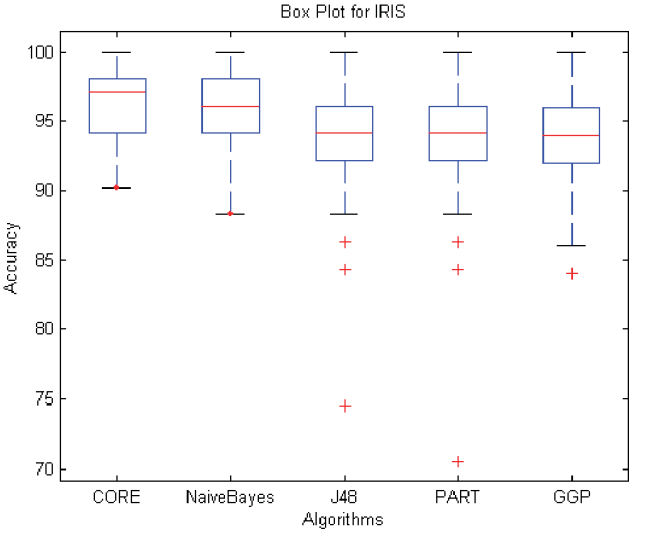


Figure 7. Box plot for the Iris dataset.

4.2.2 The medical diagnosis datasets. Medical diagnosis has been known as a crucial application domain of classification in data mining.

(A) *Experimental results*. Table 14 summarizes the classification results produced by CORE over the 100 independent runs for the medical diagnosis problems. The histograms for CORE are illustrated in figures 8–12, which again generally show a normally distributed performance. The classification rule sets that have the highest predictive accuracies on these medical datasets are presented in tables 15–19. It can be observed from the relationship between the predictive accuracy of a rule set and its rule number that rule sets with a larger number of rules will not necessarily lead to higher predictive accuracies. From the rule sets obtained, it is found that a typical rule set usually contains generalized rules followed by the more specific rules (which is particularly obvious on the Diabetes dataset). Generally, the first few rules in a rule set will cover a great portion of the samples and leave relatively few samples for the remaining rules. This situation is especially severe when there are many remaining rules. Therefore when the dataset is not noise free, a large rule number may cause

Table 14. Classification results of CORE for the medical datasets.

CORE	Hepatitis	Diabetes	Breast cancer	Sick	Heart disease
Training					
Max	95.10%	80.24%	82.98%	97.55%	88.72%
Min	77.45%	72.92%	70.21%	93.33%	77.44%
Mean	88.47%	76.96%	77.82%	95.17%	83.22%
StdDev	2.82%	1.30%	2.03%	1.18%	2.00%
Test					
Max	92.45%	80.15%	84.69%	98.05%	90.10%
Min	75.47%	69.85%	67.35%	92.67%	70.30%
Mean	84.39%	75.34%	75.41%	95.17%	80.77%
StdDev	3.72%	2.30%	3.24%	1.53%	3.17%
Avg # rules	4.14	5.99	4.32	3.49	6.24
Training time (m)	3.55	8.18	3.30	55.80	8.07

Max: Maximum classification accuracy. Min: Minimum classification accuracy. Mean: Mean classification accuracy.

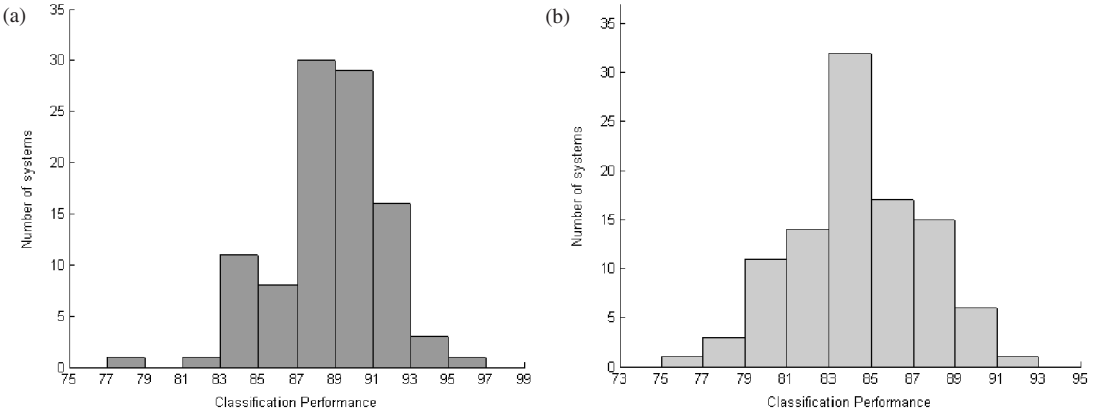


Figure 8. The performance of CORE upon the hepatitis problem: (a) training; (b) testing.

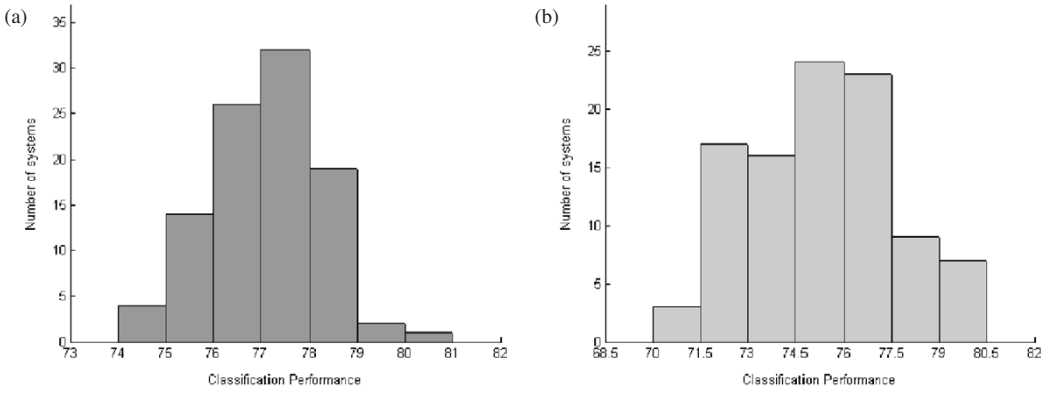


Figure 9. The performance of CORE upon the diabetes problem: (a) training; (b) testing.

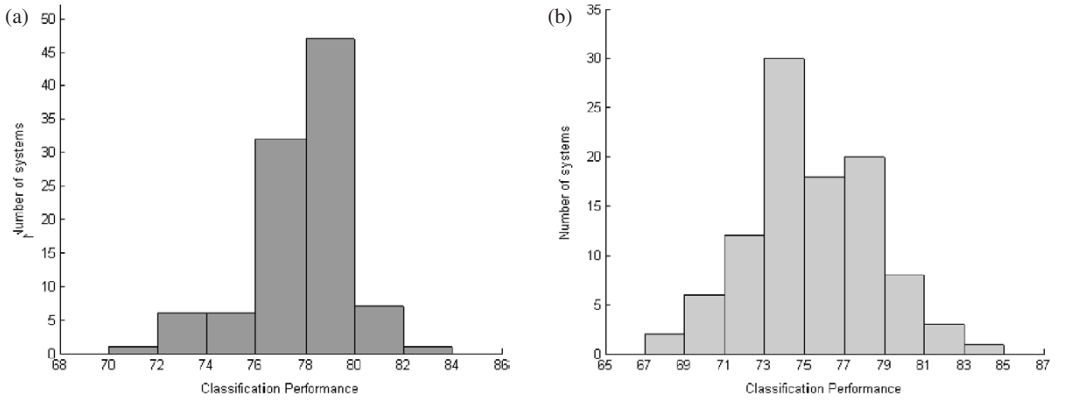


Figure 10. The performance of CORE upon the breast cancer problem: (a) training; (b) testing.

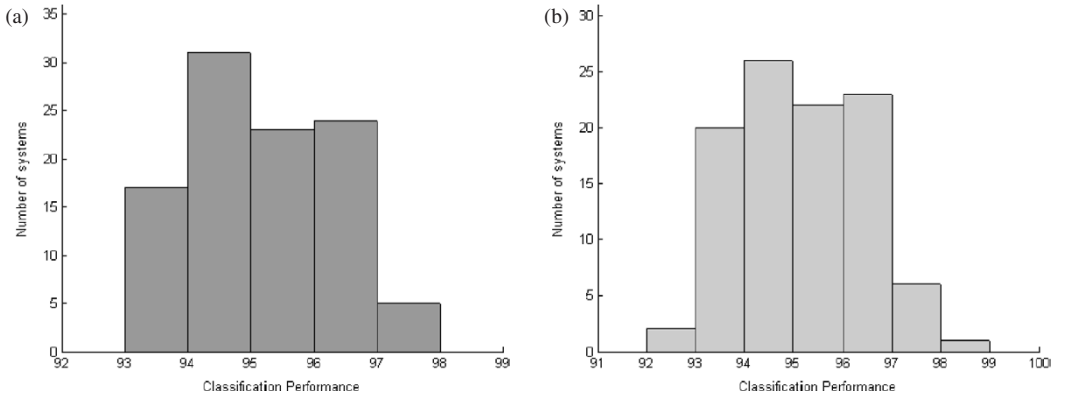


Figure 11. The performance of CORE upon the sick problem: (a) training; (b) testing.

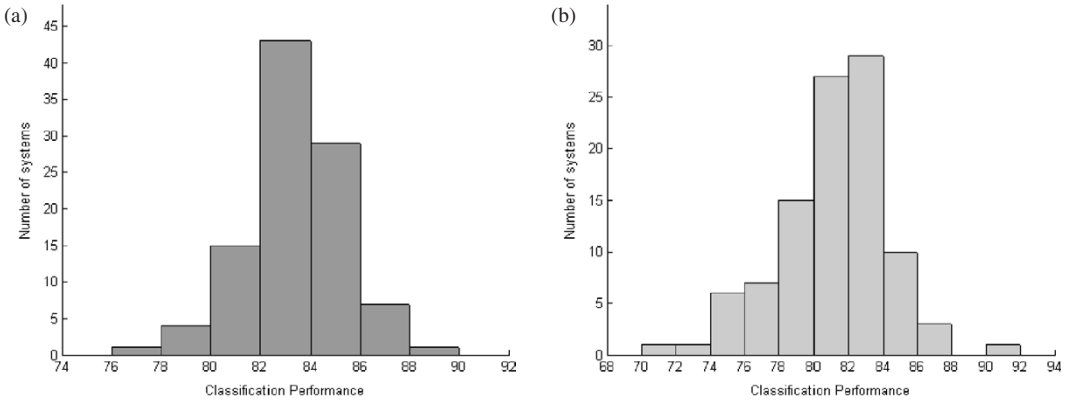


Figure 12. The performance of CORE upon the heart disease problem: (a) training; (b) testing.

Table 15. The best classification rule set of CORE for hepatitis data.

	Rule	Fitness	Support factor	Confidence factor
1	IF ASCITES = no, AND BILIRUBIN <= 1.9077, THEN Class = LIVE	1.3773	0.6373	0.9155
2	IF SPLEEN_PALPABLE != yes, AND ASCITES != yes, THEN Class = LIVE	1.2252	0.6275	0.8767
3	IF ALBUMIN <= 3.7234, THEN Class = DIE	1.0183	0.1569	0.4848
4	ELSE Class = LIVE			
Classification accuracy = 92.45%				

Table 16. The best classification rule set of CORE for diabetes data.

	Rule	Fitness	Support factor	Confidence factor
1	IF plas <= 130.7996, THEN class = tested_negative	1.0853	0.4881	0.7554
2	IF mass <> [5.8079, 29.9191], THEN class = tested_positive	0.8765	0.3202	0.4737
3	IF preg <= 4.585, AND age >< [21, 74.0987], THEN class = tested_negative	0.8966	0.4348	0.6984
4	IF plas <= 138.9471, AND insu < 538.5572, THEN class = tested_negative	1.0745	0.5237	0.7341
5	IF preg >< [0, 12.2091], AND plas >< [0, 117.3103], THEN class = tested_negative	0.9893	0.3775	0.8059
6	IF plas >< [0, 164.4243], AND age <= 77.064, THEN class = tested_negative	0.9436	0.5909	0.6765
7	IF plas > 5.8884, THEN class = tested_positive	0.6149	0.3794	0.3817
8	IF preg >= 4.195, AND plas < 124.8156, AND pres < 3.3973, AND skin <= 53.9336, AND insu >< [0, 236.2939], AND mass < 10.1695, AND pedi >< [0.078, 1.6306], AND age < 50.9739, THEN class = tested_negative	0.0128	0.004	0.6667
9	IF preg > 7.1177, AND plas >= 33.5901, AND pres <> [47.0321, 78.2014], AND mass <= 15.1642, AND age <> [61.8127, 71.4335], THEN class = tested_positive	0.0103	0.002	1.0
10	ELSE class = tested_positive			
Classification accuracy = 80.15%				

Table 17. The best classification rule set of CORE for breast cancer data.

	Rule	Fitness	Support factor	Confidence factor
1	IF age != 90–99, AND node-caps != yes, AND deg-malig != 3, THEN class = no-recurrence-events	1.0388	0.4894	0.7797
2	IF deg-malig != 3, THEN class = no-recurrence-events	1.0387	0.5426	0.7556
3	IF inv-nodes = 0–2, THEN class = no-recurrence-events	0.9924	0.5426	0.7391
4	IF deg-malig != 2, AND breast-quad != central, THEN class = recurrence-events	0.7203	0.2074	0.4063
5	ELSE class = no-recurrence-events			
Classification accuracy = 84.69%				

Table 18. The best classification rule set of CORE for sick data.

	Rule	Fitness	Support factor	Confidence factor
1	IF pregnant = f, AND T3 measured != f, AND T3 <> [0.05, 1.1936], THEN class = negative	1.3675	0.6778	0.9906
2	IF referral source != SVI, THEN class = negative	1.3492	0.7131	0.9817
3	IF on thyroxine = f, AND on antithyroid medication = f, AND tumor = f, AND TSH >< [0.005, 296.6705], AND T3 measured != f, AND T3 >< [0.05, 5.9537], AND T4U >= 0.4118, THEN class = sick	0.7905	0.0562	0.0919
4	IF T3 measured = t, AND referral source != SVHC, AND ca <> [0, 1.4164], THEN class = sick	0.0772	0.0611	0.0882
5	ELSE class = negative			
Classification accuracy = 98.05%				

over-fitting and leads to poor generalization, which is undesirable. On the other hand, having an appropriate number of specific rules could increase the precision of the rule set as a whole, which will further enhance the classification accuracy. Having considered this situation, the designers of CORE give much flexibility to its users and let them decide whether they prefer longer or shorter rule sets (users can change the number of co-populations to make the selection). If users have little knowledge of their problem, they can discard this flexibility and use the default setting of CORE, which can still generate fairly good results.

(B) *Comparisons with other works. The hepatitis dataset*. 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 in Wang *et al.* (2000), only the best result of this algorithm is compared with CORE as given in table 20. Figure 13 shows the box plot of CORE and other algorithms for comparison. As can be seen,

Table. 19. The best classification rule set of CORE for heart-C data.

	Rule	Fitness	Support factor	Confidence factor
1	IF exang != yes+, AND ca <= 0.6762, THEN num = < 50+	1.1673	0.3692	0.8276
2	IF oldpeak >< [0, 5.0737], AND thal != normal+, THEN num = > 50_1+	1.0974	0.3179	0.7294
3	IF cp = asympt+, THEN num = > 50_1+	1.0585	0.3128	0.7093
4	IF cp != non_anginal+, AND slope = up+, AND ca <> [0, 1.4164], THEN num = < 50+	0.0932	0.0256	0.7143
5	ELSE num = < 50+			
Classification accuracy = 90.10%				

Table 20. Performance comparisons for the hepatitis problem.

Algorithm	Reference	# Rules	Avg accuracy	Best accuracy	Standard deviation
CORE	–	4.14	84.40%	92.45%	3.72%
C4.5	(Quinlan 1993)	5.85	78.94	90.57%	4.84%
PART	(Frank and Witten 1998)	6.64	80.02%	94.34%	4.98%
Naïve Bayes	(John and Langley 1995)	–	83.62%	94.34%	4.90%
GA-based FKIF	(Wong *et al.* 2000)	*	–	92.9%	–
XCS	(Bernadó *et al.* 2001)	–	76.8%	–	–

*The authors did not provide average rule number of the rule sets generated by their algorithm.

although CORE does not achieve the best accuracy for this dataset, it does produce the best average accuracy with no outlier data point.

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 21 for comparison. 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 21. CORE 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. Figure 14 shows the comparisons of CORE with other algorithms using the box plot. Clearly, CORE has

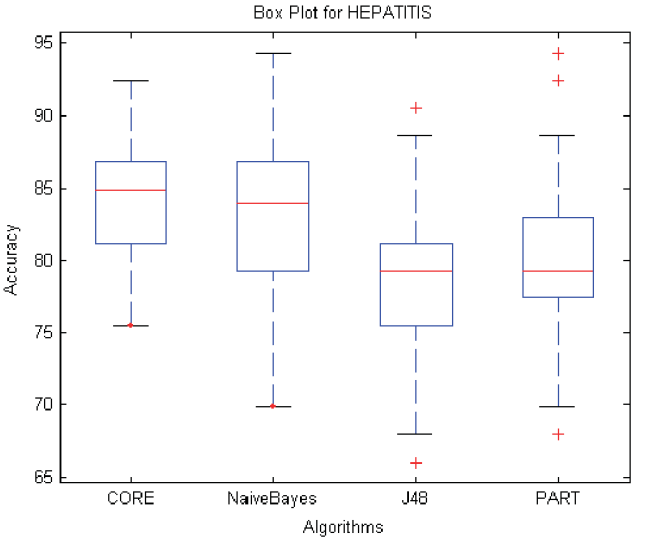


Figure 13. Box plot for the hepatitis problem.

Table 21. Performance comparisons for the diabetes problem.

Algorithm	Reference	# Rules	Avg accuracy	Best accuracy	Standard deviation
CORE	–	5.99	75.34%	80.15%	2.30%
C4.5	(Quinlan 1993)	5.85	73.13%	77.39%	2.55%
PART	(Frank and Witten 1998)	6.64	72.78%	80.08%	2.53%
Naïve Bayes	(John and Langley 1995)	–	75.09%	81.61%	2.45%
Itrule	(Michie *et al.* 1994)	–	75.5%	–	–
CN2	(Michie *et al.* 1994)	–	71.1%	–	–
CART	(Michie *et al.* 1994)	–	74.5%	–	–
AC^2	(Michie *et al.* 1994)	–	72.4%	–	–
Cal5	(Michie *et al.* 1994)	–	75.0%	–	–
GGP	(Wong 2001)	14.54	72.60%	77.95%	2.97%
XCS	(Bernadó *et al.* 2001)	–	75.4%	–	4.7%
XCSR	(Bacardit and Garrell 2003)	3.4	74.2%	–	–

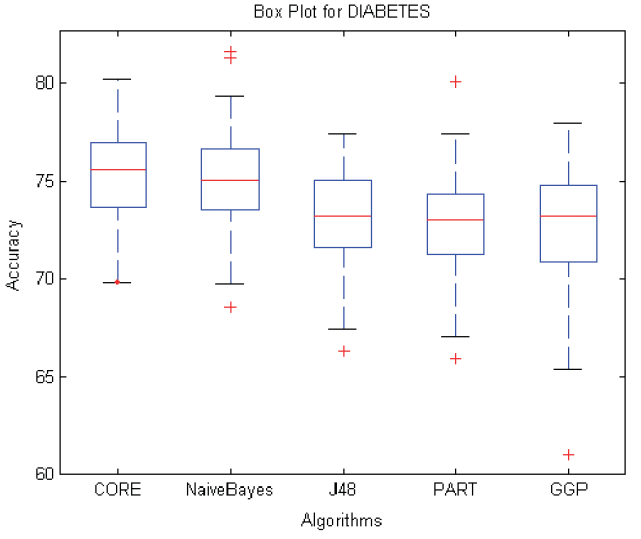


Figure 14. Box plot for the diabetes problem.

achieved the best performance with very competitive classification results.

The breast cancer dataset: Table 22 compares the classification results from CORE, C4.5 rules, PART and Naïve Bayes for the Breast Cancer dataset. In terms of concision, the average rule number in a rule set generated by CORE is less than one third of those obtained by C4.5 rules and PART. Figure 15 shows the comparison results of CORE with other algorithms using the box plot. As can be seen, CORE has again achieved a very competitive classification result for this dataset.

The sick dataset: The comparison results of the sick dataset are summarized in table 23. C4.5 and PART generated very good results on this dataset in terms of classification accuracy. The box plot in figure 16 shows that CORE is less accurate for this dataset compared to other rule-based classifiers although it has outperformed Naïve Bayes with great confidence. On the other hand, the rule sets from CORE on this dataset are fairly succinct and have less than 4 rules on average. The C4.5 and PART have approximately 21 and 17 rules respectively, which are all more than the permitted number of rules in the rule sets from CORE. Although the small number of rules in the rule sets of CORE is relatively easy to understand for users, it also accounts for the slightly worse classification accuracy of CORE on this dataset, since less rules often result in the loss of precision in the classification. To overcome the problem, users can increase the permitted number of rules in CORE at the expense of longer training time by adding more co-populations, which will enhance the appearance opportunity of the longer rule sets in the final results.

The heart disease 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. As shown in table 24, CORE obtains more accurate and concise rule sets than C4.5 rules and PART for this dataset. However, the average accuracy obtained by Naïve

Table 22. Performance comparisons for the breast cancer problem.

Algorithm	Reference	# Rules	Avg accuracy	Best accuracy	Standard deviation
CORE	–	4.32	75.41%	84.69%	3.24%
C4.5	(Quinlan 1993)	14.27	71.81%	78.35%	3.55%
PART	(Frank and Witten 1998)	13.71	69.32%	80.41%	4.33%
Naïve Bayes	(John and Langley 1995)	–	72.34%	94.34%	3.29%

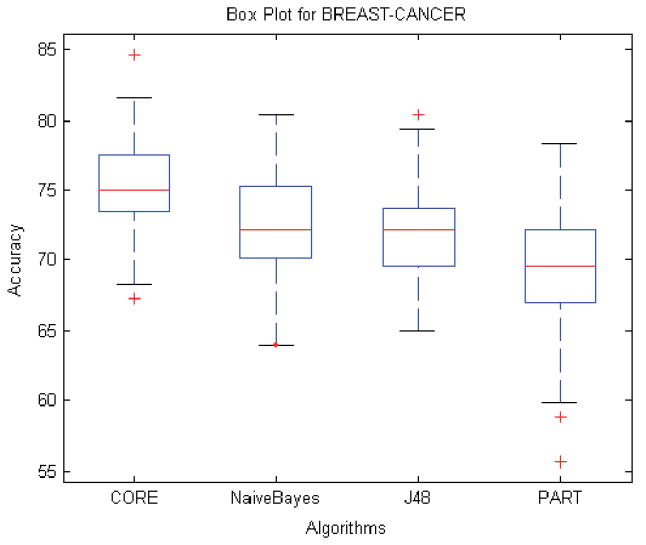


Figure 15. Box plot for the breast cancer problem.

Bayes classifier is better than CORE, which may be due to the distribution of this dataset that benefits the statistical type of classifiers. It is believed that the Naïve Bayes is an optimal classifier when the class distributions and *a priori* probabilities are known. Figure 17 shows the comparison results of CORE with other algorithms using the box plot. In this case, the Naïve Bayes classifier offers the best accuracy although CORE has the lowest standard deviation among all algorithms. If only rule-based classifiers are considered, however, CORE achieved the best results by outperforming C4.5 rules and PART significantly.

4.2.3 The financial dataset. (A) *Experimental results*. Table 25 summarizes the average results of CORE over 100 simulation runs, while the classification rule set that has the highest predictive accuracy of this data is shown in table 26. The histograms are given in figure 18. As can be seen from the training performance histograms, there are over 30 runs with accuracy in the range of 87% to 87.5% and 17 runs with accuracy in the range of 85% to 85.5% over the total 100 simulation runs. Such a phenomenon has caused the overall histograms to not be normally distributed (the same phenomenon also applied to the Hepatitis dataset). The reason for this phenomenon is that the evolutionary search has been trapped in local optima during these simulation runs, which could be overcome by having a larger population and generation size. To further examine the classification results, the rule number of the best rule set in every evolutionary run is recorded in table 27. As can be seen, although the rule number varies from 1 to 16, rule sets with number of rules below 5 cover over fifty percent of all the best rule sets in 100 runs. This shows that short rule sets have a good chance of survival in the course of co-evolution, which justifies their excellent classification performances for this dataset.

(B) *Comparisons with other works*. As shown in table 28 and figure 19, CORE has offered a very good classification solution for the Credit Card Assessment dataset. Setiono and Liu (1997) also applied the algorithm of NeuroLinear to this dataset and generated a concise rule set with an average rule number of 6.60. As shown in the table, the Naïve Bayes classifier has the worst performance for this dataset compared to other algorithms.

5. Discussions

The proposed CORE has been examined on seven datasets obtained from UCI machine learning repository and has produced good classification results as compared to many existing classifiers. Most of the comparisons were performed statistically using box plots to show the robustness of the proposed classifier. Extensive simulation results show that CORE has

Table 23. Performance comparisons for the Sick problem.

Algorithm	Reference	# Rules	Avg accuracy	Best accuracy	Standard deviation
CORE	–	3.49	95.17%	98.05%	1.53%
C4.5	(Quinlan 1993)	21.61	98.59%	99.53%	0.12%
PART	(Frank and Witten 1998)	17.05	98.27%	99.06%	0.16%
NaïveBayes	(John and Langley 1995)	–	92.59%	94.93%	1.28%

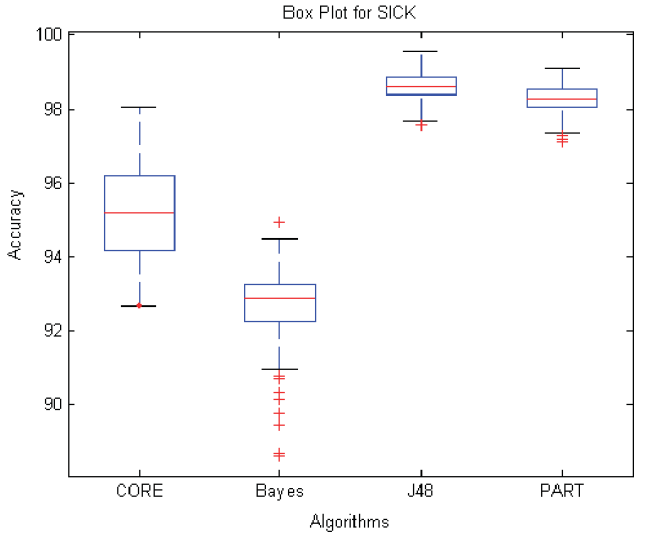


Figure 16. Box plot for the Sick problem.

outperformed two other rule-based classifiers (C4.5 and PART) in almost all test problems except for the Sick dataset and is very competitive as compared to statistical based techniques, such as Naïve Bayes (e.g. Naïve Bayes achieved better results only on the Heart-C problem). One reason for CORE being less efficient than the C4.5 and PART in terms of generalization ability for Sick dataset could be due to the low number of rules used. While CORE uses 3.49 (on average) number of rules, C4.5 uses 21.61 number of rules and PART uses 17.05 number of rules, these figures are about 600% and 500% more compared to the number of rules CORE uses. In order to improve the performance of CORE for this problem, the upper limit of allowed rules can be increased but at the expense of higher computational cost. The performance comparisons to other evolutionary based classifiers (GP-Co, GGP, GBML, GPCE, GA-based FKIF, XCS and XCSR) 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 have been 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 CORE by comparing it to some established methods widely used in the literature.

The paired *t*-tests with Bonferroni correction made for multiple comparisons have been performed between CORE and three widely used classifiers (C4.5 rules, PART and Naïve Bayes) by taking each dataset as independent observation and the results are shown in table 29. These *p*-values suggested that there is not much significant difference between the means as opposed to the results presented in previous tables; this could be due to the number of independent observations taken being too small. The results reported by CORE shown in previous tables show the lowest standard deviation in all the problems except Sick, compared to all other algorithms (subjected to availability of standard deviation value). The box plots show that CORE has a relatively lesser number of outliers as compared to traditional rule-based classifiers (C4.5 rule and PART), which indicates that CORE is relatively more robust and less affected by the random partition of learning and testing sets.

Observations from results show that larger rule sets might not perform better than smaller rule sets in terms of classification accuracy on the training set. This observation opposes findings that there is a trade off between the classification accuracy on the training set and the number of rules used (Ishibuchi and Yamamoto 2003, Ishibuchi and Namba 2004). One of the reasons might be that in large rule sets, it does not necessary imply that all rules in the rule set are considered. Very often, an instance will only visit the first few rules and the later rules are not considered. Therefore in such cases, it does not really matter whether the rule set is

Table 24. Performance comparisons for the heart disease problem.

Algorithm	Reference	# Rules	Avg accuracy	Best accuracy	Standard deviation
CORE	–	6.24	80.77%	90.10%	3.17%
C4.5	(Quinlan 1993)	18.12	76.61%	84.16%	3.27%
PART	(Frank and Witten 1998)	15.13	77.97%	86.14%	3.65%
NaïveBayes	(John and Langley 1995)	–	82.96%	90.10%	3.37%
NeuralLinear	(Setiono and Liu 1997)	5.69	78.15%	–	6.86%
XCS	(Bernadó *et al.* 2001)	–	80.3%	–	7.8%

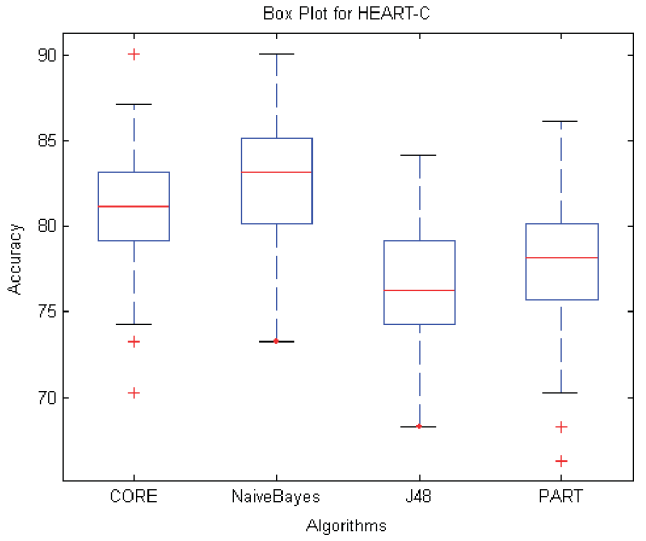


Figure 17. Box plot for the heart-C problem.

extremely big as the latter rules have become redundant. This type of large rule set actually has the same effect as small rule sets whereby their classification accuracy does not outperform that of the small rule sets. It can also be observed from the experiment results that the number of rules for the best rule set produced by CORE is relatively small compared to other algorithms. For all the problems, the size of rule sets (average) is significantly smaller or comparable to other rule based algorithms. This is an important advantage of CORE since the comprehensibility of the classification results is directly reflected by its number of rules.

Figure 20 shows the convergence performance analysis of CORE for all the datasets. As can be seen, although the main population evolves in a very stochastic way (mainly due to the regenerate operator and to a lesser extent by the mutation operator), it provides a ground for the co-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 good solutions. The stochastic nature of the main population plays an important role in the proposed coevolutionary model to maintain the diversity of the individual pool. As shown in figure 20, the convergence of CORE is fast and with less than 40 generations on average. The good performance in terms of generalization ability and fast convergence could be suggested by the large solution space visited by the algorithm. With regards to the small number of evaluations (co-population size = 50 and converges around 40 generations on average), this might suggest that a simpler, more computationally efficient local search technique, e.g. hill climbing technique, could also perform relatively well. However local search techniques are characterized by getting trapped in local optima, where solutions found are usually sub-optimal points and do not exhibit good generalization ability.

6. Conclusions

This paper has proposed a cooperative coevolutionary algorithm (CORE) for rule extraction and classification in data mining applications. CORE utilizes the evolutionary algorithm principles which possess global search ability to search for rules and rule sets. These solutions are presented in high level linguistic rule sets that are easily comprehensible for humans. Unlike existing evolutionary approaches, the proposed coevolutionary classifier coevolves the rules and rule sets concurrently in two cooperative populations. The main population encodes a population of rules using Michigan encoding which are syntactically shorter thus making the search for good candidate solutions faster. The co-populations

Table 25. Summary of results for the credit card assessment dataset.

	Max	Min	Mean	StdDev	Avg # rules	Training time (m)
Training	88.79%	84.18%	86.71%	1.00%	4.75	11.03
Test	91.49%	81.70%	86.49%	1.86%	4.75	11.03

Max: Maximum classification accuracy. Min: Minimum classification accuracy. Mean: Mean classification accuracy.

Table 26. Best classification rule set for the credit-A data.

	Rule	Fitness	Support factor	Confidence factor
1	IF A9 != t, AND A15 < 36765.7028, THEN class = −	1.4402	0.4352	0.9252
2	IF A4 != l, AND A7 != ff, AND A10 = t, AND A15 > < [0, 42794.9549], THEN class = +	1.0532	0.2813	0.7314
3	IF A4 != y, AND A13 != s, AND A14 < 1031.244, THEN class = +	0.8579	0.3429	0.5235
4	ELSE class = −			
Classification accuracy = 91.06%				

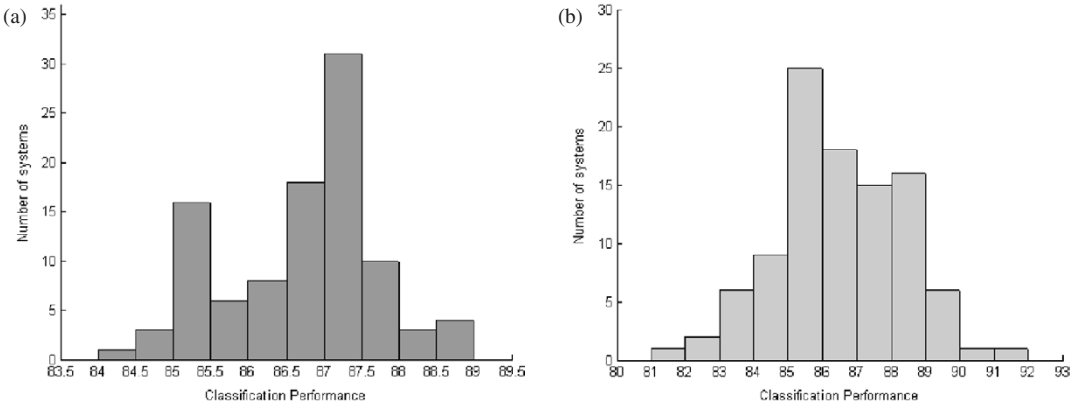


Figure 18. Performance of CORE for the credit card assessment problem: (a) training; (b) testing.

are then presented with these good candidate rules to form rule sets using Pittsburgh encoding, and in this way the normally large search space often encountered while using Pittsburgh encoding is confined. The rule sets in the co-population take into account the rules interaction in order to produce good solutions, and this is not achievable in the main population. Rules and rule sets are coevolved simultaneously in CORE to ensure that the solutions are cohesive and comprehensive. CORE ensures that the merits of both encoding approaches are

Table 27. Occurrence of different number of rules.

Rule set	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16
Appearance time in 100 runs	11	9	6	20	13	7	6	5	7	3	5	3	1	1	3

Table 28. Performance comparisons for the credit card assessment problem.

Algorithm	Reference	# Rules	Avg accuracy	Best accuracy	Standard deviation
CORE	–	4.75	86.48%	91.49%	1.86%
C4.5	(Quinlan 1993)	19.34	85.14%	90.21%	2.06%
PART	(Frank and Witten 1998)	27	83.59%	89.36%	2.25%
Naïve Bayes	(John and Langley 1995)	–	77.87%	83.83%	2.57%
Itrule	(Michie *et al.* 1994)	–	86.3%	–	–
CN2	(Michie *et al.* 1994)	–	79.6%	–	–
CART	(Michie *et al.* 1994)	–	85.5%	–	–
AC^2	(Michie *et al.* 1994)	–	81.9%	–	–
Cal5	(Michie *et al.* 1994)	–	86.9%	–	–
Neura Linear	(Setiono and Liu 1997)	6.60	83.64%	–	5.74%
XCS	(Bernadó *et al.* 2001)	–	84.8%	–	–

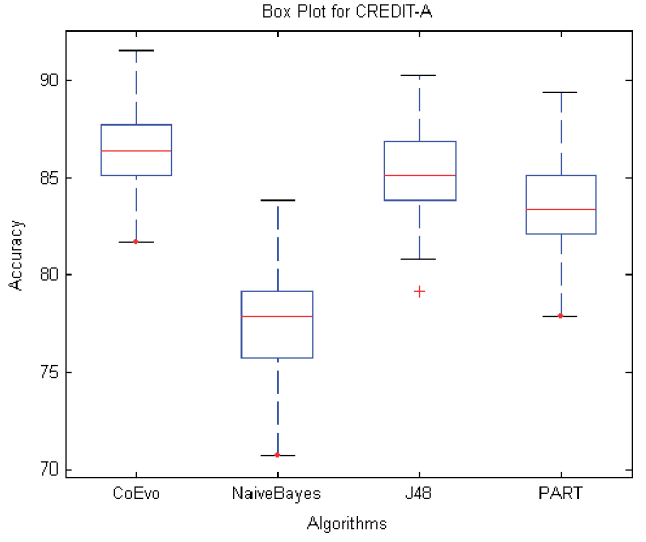
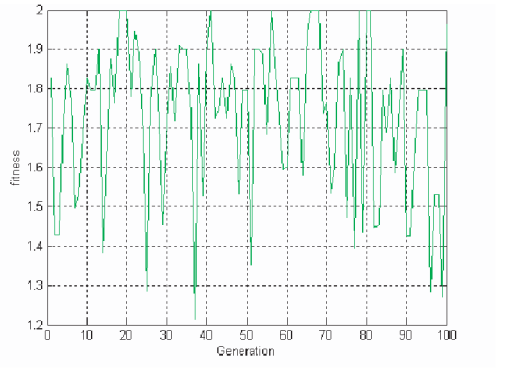


Figure 19. Box plot for the credit-A problem.

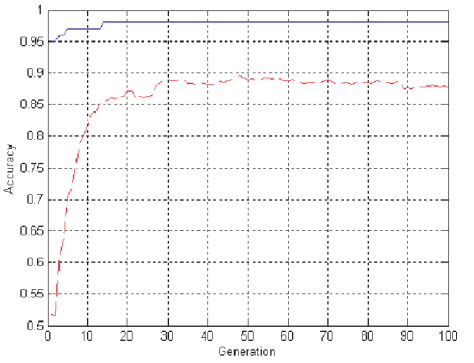
Table 29. Paired *t*-test.

	C4.5	PART	Naïve Bayes
P-value	0.037532	0.025068	0.080511

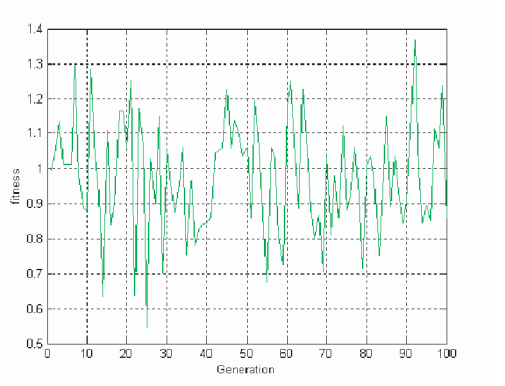
utilized while compromising their drawbacks. The proposed CORE has been extensively validated upon seven datasets obtained from the UCI Machine Learning Repository, and the results have been analysed both qualitatively and statistically. Comparison results show that the proposed CORE produces comprehensive and good classification rules for all the datasets, which are very competitive or better than many classifiers widely used in literature. For most datasets, CORE is able to produce a small number of rules for the best rule set without the expense of generalization ability. Moreover, the standard deviation reported by CORE is the smallest in almost all cases. Simulation results obtained from the box plots have unveiled that CORE is relatively robust and invariant to random partitioning of datasets. CORE appears as a good rule-based classifier for two-class and multi-class problems. To reduce the computational effort significantly, the CORE is currently being integrated into the "Paladin-DEC" distributed evolutionary computing framework (Tan *et al.* 2002b), where multiple inter-communicating subpopulations are implemented to share and distribute the classification workload among multiple computers over the Internet. Other distributed frameworks can also be implemented to test the efficiency.



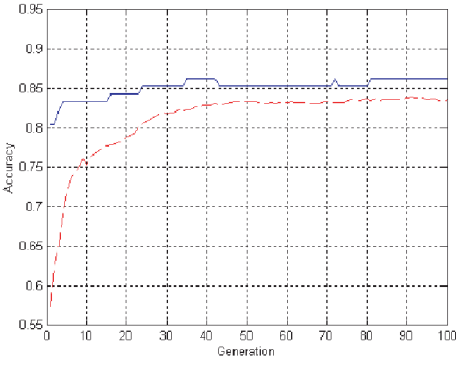
Iris (a)



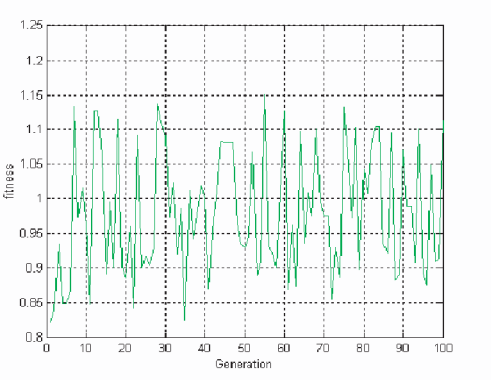
Iris (b)



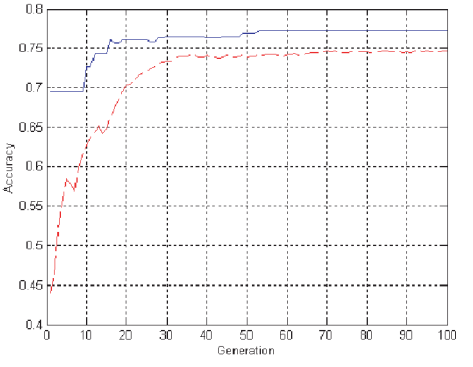
Hepatitis (a)



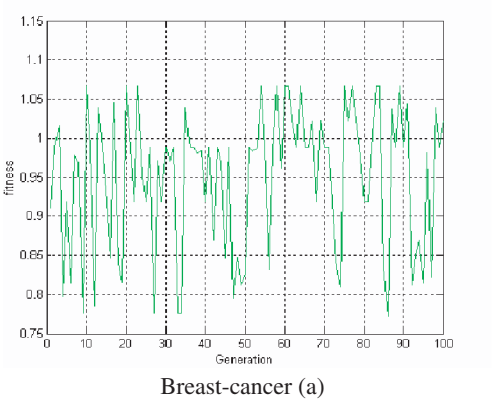
Hepatitis (b)



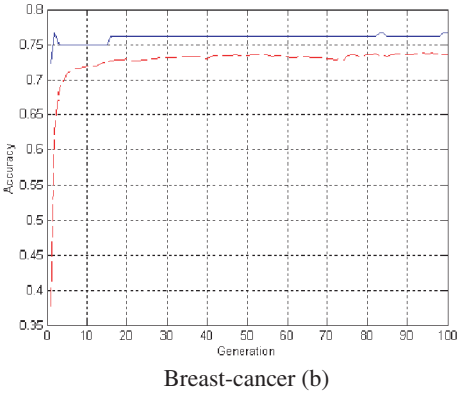
Diabetes (a)



Diabetes (b)

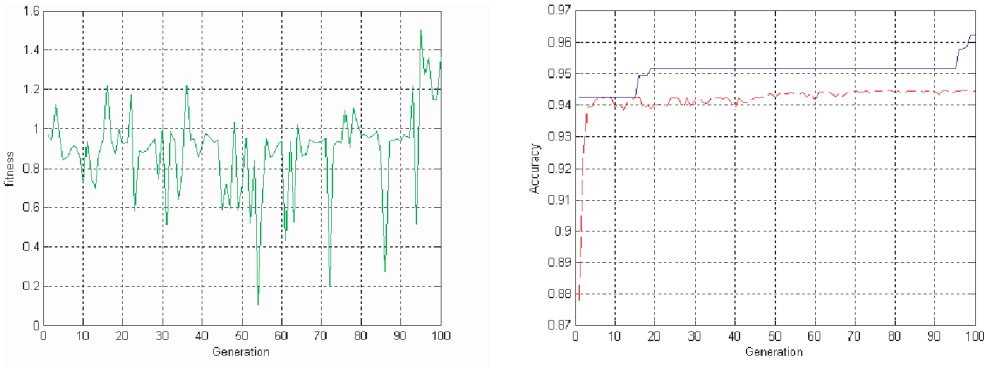


Breast-cancer (a)

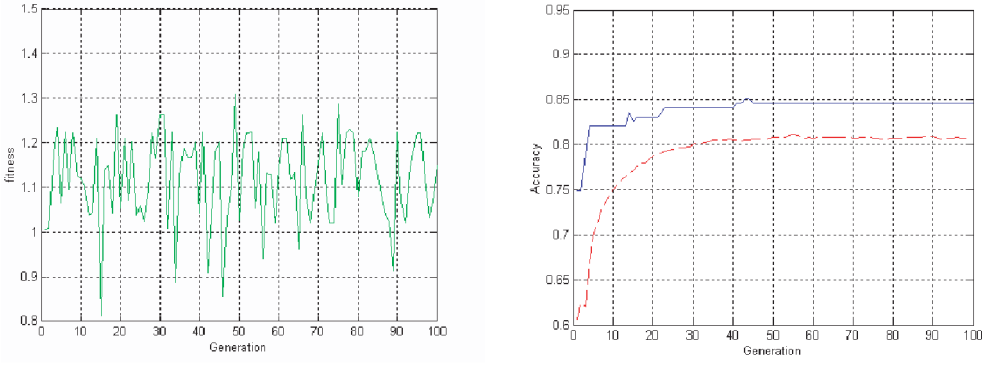


Breast-cancer (b)

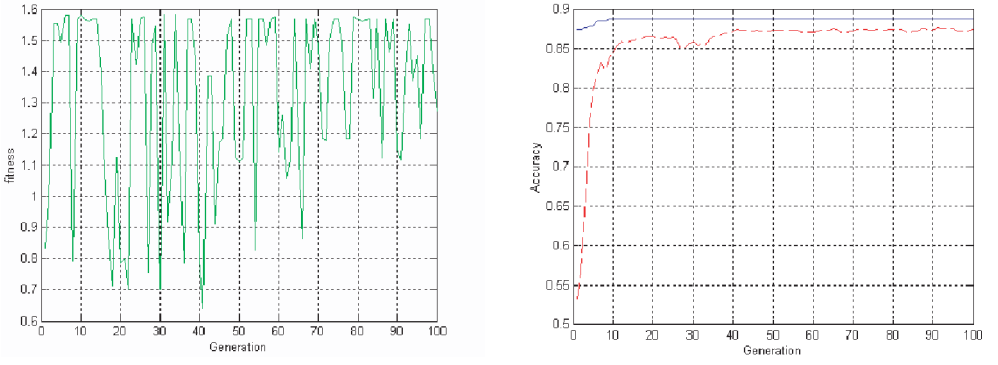
Figure 20. Coevolution progress for all datasets: (a) fitness of main population; (b) fitness of co-populations.



Sick (a) Sick (b)



Heart-c (a) Heart-c (b)



Credit-A (a) Credit-A (b)

- - - average classification accuracy of co-populations on training data

— best classification accuracy of co-populations on training data

Figure 20. Continued.

Acknowledgement

The authors would like to thank Dr. Wong Man Leung for providing the source code of GGP for comparisons and Dr. Kin Yee Chan for sharing her knowledge on statistical paired *t*-tests.

References

P.J. Angeline and J.B. Pollack, "Competitive environments evolve better solutions for complex tasks", in *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana-Champaign, IL, USA, 1993, pp. 264–270.

J. Bacardit and J.M. Garrrell, ''Evolving multiple discretizations with adaptive intervals for a Pittsburgh rule-based learning classifier system'', in *Proceedings of the Genetic and Evolutionary Computation Conference*, (LNCS 2724), Berlin: Springer-Verlag, 2003, pp. 1818–1831.

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

A. Barry, J. Holmes and X. Llorà, ''Data mining using learning classifier systems'', *Appl. Learn. Classif. Sys.*, 150, pp. 15–67, 2004.

E. Bernadó, X. Llorà, and J.M. Garrell, ''XCS and GALE: a comparative study of two learning classifier systems on data mining'', in *Advances in Learning Classifier Systems*, P. Lanzi, W. Stolzmann and S. Wilson, Eds, *4th International Workshop*, volume 2321 of *Lecture Notes in Artificial Intelligence*, New York: Springer, 2001, pp. 115–132.

C.C. Bojarczuk, H.S. Lopes and A.A. Freitas, ''Genetic programming for knowledge discovery in chest-pain diagnosis'', *IEEE Eng. Med. Biol. Mag.*, 4, pp. 38–4, 2000.

M. Brameier and W. Banzhaf, ''A comparison of linear genetic programming neural networks in medical data mining'', *IEEE Trans. Evolut. Comp.*, 5, pp. 17–26, 2001.

D.R. Carvalho and A.A. Freitas, ''A genetic algorithm for discovering small disjunct rules in data mining'', *Appl. Soft Comp.*, 2, pp. 75–88, 2002.

D.R. Carvalho and A.A. Freitas, ''A hybrid decision tree/genetic algorithm method for data mining'', *Inform. Sci.*, 63, pp. 77–86, 2004.

R. Cattral, F. Oppacher and D. Deugo, ''Rule acquisition with a genetic algorithm'', in *Proceedings of the 1999 Congress on Evolutionary Computation*, Vol. 1, Washington DC, 1999, pp. 125–129.

G. Cestnik, I. Konenenko and I. Bratko, ''Assistant-86: a knowledge-elicitation tool for sophisticated users'', in *Machine Learning, Sigma Press*, Bratko, L., Ed., Imprint: Belmont, Calif: Wadsworth International Group, Boston: Duxburg Press, 1983, pp. 31–45 .

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

C.B. Congdon, ''Classification of epidemiological data: a comparison of genetic algorithm and decision tree approaches'', in *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, La Jolla, CA, USA, 2000, pp. 442–449.

I. De Falco, A. Della Cioppa and E. Tarantino, ''Discovering interesting classification rules with genetic programming'', *Appl. Soft Comp.*, 23, pp. 1–13, 2002.

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

U. Fayyad, ''Data mining and knowledge discovery in databases: implications for scientific databases'', in *Proceedings of the Ninth International Conference on Scientific and Statistical Database Management*, Olympia, WA, USA, 1997, pp. 2–11.

M.V. Fidelis, H.S. Lopes and A.A. Freitas, ''Discovering comprehensible classification rules with a genetic algorithm'', in *Proceedings of IEEE Congress on Evolutionary Computation*, Vol. 1, La Jolla, CA, USA, 2000, pp. 805–810.

R.A. Fisher, ''The use of multiple measurements in taxonomic problems'', *Ann. Eugen.*, 7, pp. 179–188, 1936.

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

A.A. Freitas, ''Understanding the crucial role of attribute interaction in data mining'', *Artif. Intellig. Rev.*, 16, pp. 177–199, 2001.

A.A. Freitas, ''A survey of evolutionary algorithms for data mining and knowledge discovery'', in *Advances in Evolutionary Computation*, A. Ghosh and S. Tsutsui, Eds, Berlin: Springer-Verlag, 2002a, pp. 819–845.

A.A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, A. Ghosh and S. Tsutsui, Eds, Berlin: Springer-Verlag, 2002b.

E.R. Hruschka and N.F.F. Ebecken, ''A clustering genetic algorithm for extracting rules from supervised neural network models in data mining tasks'', *Inter. J. Comp., Syst. Sig.*, 1, pp. 17–29, 2000.

H. Ishibuchi and S. Namba, ''Evolutionary multiobjective knowledge extraction for high-dimensional pattern classification problems'', in *Lecture Notes in Computer Science 3242: Parallel Problem Solving from Nature – PPSN VIII*, Berlin: Springer, 2004, pp. 1123–1132.

H. Ishibuchi and T. Yamamoto, Effects of three-objective genetic rule selection on the generalization ability of fuzzy rule-based systems, in *Lecture Notes in Computer Science 2632: Evolutionary Multi-Criterion Optimization*, Berlin: Springer, 2003, pp. 608–622.

H. Ishibuchi and T. Yamamoto, ''Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining'', *Fuzzy Sets Sys.*, 141, pp. 59–88, 2004.

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

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

V. Khare, X. Yao and B. Sendhoff, ''Credit assignment among neurons in co-evolving populations'', in *Proceedings of the Parallel Problem Solving from Nature VIII Conference*, Birmingham, UK, 2004 pp. 882–891.

J.K. Kishore, L.M. Patnaik, V. Mani and V.K. Agrawal, ''Application of genetic programming for multicategory pattern classification'', *IEEE Trans. Evolut. Comp.*, 4, pp. 242–258, 2000.

P.L. Lanzi and M. Colombetti, ''An extension to the XCS classifier system for stochastic environments'', in *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA: Morgan Kaufmann, 1999, pp. 353–360.

P.L. Lanzi and A. Perrucci, ''Extending the representation of classifier conditions, part II: from messy coding to s-expression'', in *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA: Morgan Kaufmann, 1999, pp. 345–352.

R.J.A. Little and D.B. Rubin, *Statistical Analysis with Missing Data*, New York: Wiley, 1987.

H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, Boston: Kluwer Academic Publishers, 1998.

R.R.F. Mendes, F.B. Voznika, A.A. Freitas and J.C. Nievola, ''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*, New York: Springer-Verlag, 2001, pp. 314–325.

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

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

T.M. Mitchell, *Machine Learning*, New York: McGraw Hill, 1997.

D.E. Moriarty, ''Symbiotic evolution of neural networks in sequential decision tasks''. PhD thesis, The University of Texas at Austin (1997).

E. Noda, A.A. Freitas and H.S. Lopes, ''Discovering interesting prediction rules with a genetic algorithm'', in *Proceedings of the IEEE Conference on Evolutionary Computation*, Washington DC, 1999, pp. 1322–1329.

C.A. Peña-Reyes and M. Sipper, ''Fuzzy CoCo: a cooperative-coevolutionary approach to fuzzy modelling'', *IEEE Trans Fuzzy Sys.*, 9, pp. 727–737, 2001.

M.A. Potter and K.A. De Jong, ''A cooperative coevolutionary approach to function optimization'', in *Proceedings of the Parallel Problem Solving from Nature III Conference*, Jerusalem, Israel, 1994, pp. 249–257.

M.A. Potter and K.A. De Jong, ''Cooperative coevolution: an architecture for evolving coadapted subcomponents'', *Evolut. Comp.*, 8, pp. 129, 2000.

A.R. Pozo and M. Hasse, "A genetic classifier tool", in *Proceedings of the 20th International Conference of the Chilean Computer Science Society*, Santiago, Chile, 2000, pp. 14–23.
L. Prechelt, "Some notes on neural learning algorithm benchmarking", *Neuralcomputing*, 9, pp. 343–347, 1995.
J.R. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.
C.D. Rosin and R.K. Belew, "New methods for competitive coevolution", *Evolut. Comp.*, 5, pp. 1–29, 1997.
S.E. Rouwhorst and A.P. Engelbrecht, "Searching the forest: using decision trees as building blocks for evolutionary search in classification databases", in *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, California, CA, USA, 2000, pp. 633–638.
S. Saxon and A. Barry, "XCS and the monk's problems", *Learning Classifier Systems, From Foundations to Applications Lecture Notes in Artificial Intelligence (LNAI-1813)*, P.L. Lanzi W. Stolzmann and S.W. Wilson, Eds, Berlin: Springer-Verlag, 2000, pp. 223–242.
R. Setiono and H. Liu, "NeuroLinear: from neural networks to oblique decision rules", *Neurocomputing*, 17, pp. 1–24, 1997.
K.C. Tan, A. Tay, T.H. Lee and C.M. Heng, "Mining multiple comprehensible classification rules using genetic programming", in *Proceedings of the IEEE Congress on Evolutionary Computation*, Honolulu, Hawaii, 2002a, pp. 1302–1307.
K.C. Tan, T.H. Lee and E.F. Khor, "Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons", *Artif. Intellig. Rev.*, 17, pp. 251–290, 2002b.
K.C. Tan, Q. Yu, C.M. Heng and T.H. Lee, "Evolutionary computing for knowledge discovery in medical diagnosis", *Artif. Intellig. Med.*, 27, pp. 129–154, 2003.
K.C. Tan, Q. Yu and J.H. Ang, "A dual-objective evolutionary algorithm for rules extraction in data mining", *Comput. Optim. Appl.*, 34, pp. 273–294, 2006.
C.H. Wang, T.P. Hong and S.S. Tseng, "Integrating membership functions and fuzzy rule sets from multiple knowledge sources", *Fuzzy Sets Sys.*, 112, pp. 141–154, 2000.
S.W. Wilson, "Classifier fitness based on accuracy", *Evolut. Comp.*, 3, pp. 149–175, 1995.
S.W. Wilson, "Get real! XCS with continuous-valued inputs", *Learning Classifier Systems. From Foundations to Applications Lecture Notes in Artificial Intelligence (LNAI-1813)*, P.L. Lanzi, W. Stolzmann and S.W. Wilson, Eds, Berlin: Springer-Verlag, 2000, pp. 209–222.
I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, CA: Morgan Kaufmann Publishers, 1999.
M.L. Wong and K.S. Leung, *Data Mining Using Grammar Based Genetic Programming and Applications*, London: Kluwer Academic Publishers, 2000.
M.L. Wong, "A flexible knowledge discovery system using genetic programming and logic grammars", *Dec. Sup. Syst.*, 31, pp. 405–428, 2001.
X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks", *IEEE Trans. Neural. Netw.*, 8, pp. 694–713, 1997.

K. C. Tan received the B.Eng. degree with first class honors in electronics and electrical engineering and the PhD degree from the University of Glasgow, Glasgow, Scotland, in 1994 and 1997, respectively. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, National University of Singapore. Dr. Tan has authored or coauthored more than 130 journals and conference publications and has served as a program committee or organizing member for many international conferences. He currently holds Associate Editor appointment in *IEEE Transactions on Evolutionary Computation* and *International Journal of Systems Science*. His current research interests include computational intelligence, evolutionary computing, and engineering design optimization.

Q. Yu received the B.E. degree from Zhejiang University, Hangzhou, China, in 2001 and the M.E. degree in electrical and computer engineering from National University of Singapore, Singapore, in 2003. He is currently pursuing his Ph.D. degree in computer science at Virginia Tech, Blacksburg, Virginia. He is a research assistant in the E-Commerce and E-Government Laboratory in Virginia Tech. His research interests include several areas in databases, with particular emphasis on Web service query optimization and rule-based data classification.

J. H. Ang received the B.Eng. degree in Electrical Engineering from the National University of Singapore, Singapore, in 2004. He is currently pursuing his PhD degree at the Centre for Intelligent Control, National University of Singapore. His research interest centers on applying computational intelligence techniques for data mining applications.