ELSEVIER

Computers in Industry 46 (2001) 95–106

COMPUTERS IN INDUSTRY

www.elsevier.com/locate/compind

A prototype genetic algorithm-enhanced rough set-based rule induction system

Li-Pheng Khoo*, Lian-Yin Zhai

School of Mechanical and Production Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

Received 6 September 2000; accepted 19 May 2001

Abstract

The ability to learn from empirical data or the observation of a real world and accurately predict future instances is an important feature expected in human. It allows knowledge to be gained by experience and decision rules induced from empirical data. One of the major obstacles in performing rule induction from empirical data is the inconsistency of information about a problem domain. Rough set theory provides a novel way of dealing with vagueness and uncertainty. When coupled with genetic algorithms, a rule induction engine that is able to induce probable rules from inconsistent information can possibly be developed. This paper presents an integrated approach that combines rough set theory, genetic algorithms and Boolean algebra, for inductive learning. Using such an approach, a prototype system (RClass-Plus) that discovers rules from inconsistent empirical data, has been developed. The system was validated using the data obtained from a case study. The results of the validation are presented.

Keywords: Rule induction; Rough sets; Genetic algorithms

1. Introduction

Decision support is based on human knowledge about a specific part of a real or abstract world. If the knowledge is gained by experience, decision rules can possibly be induced from the empirical data obtained. The ability of acquiring decision rules from the environment, that is, the empirical data, is an important requirement for both natural and artificial organisms. In artificial intelligence, such ability can be acquired by performing inductive learning [1]. Many techniques such as decision tree learning [2], and genetic algorithm-based learning [3], have been developed to carry out inductive learning. Decision tree learning technique employs a decision tree to achieve the learning function. Using such an approach, Quinlan developed the Inductive Dichotomizer 3 (ID3) system in 1986, and its later versions C4.5 [2] and C5.0 in 1992 and 1997, respectively. Genetic algorithm-based learning technique, on the other hand, takes advantage of the unique search engine of genetic algorithms (GAs) to glean probable decision rules from its search space.

Knowledge has the nature of granularity and may be incomplete, imprecise, or even conflicting. For example, the opinion about the performance of an engine that is perceived by two engineers could be different. Furthermore, some notions can only be defined vaguely as they cannot be precisely and accurately described. For example, if the gas emitted from an

* Corresponding author. Tel.: +65-790-55-98; fax: +65-792-40-62.
E-mail address: mlpkhoo@ntu.edu.sg (L.-P. Khoo).

0166-3615/01/$ – see front matter
PII: S0166-3615(01)00117-8

engine is *smoky*, it is likely that *incomplete combustion* has taken place. Words such as *smoky and incomplete combustion* are vague, imprecise and are difficult to quantify. As a result, inductive learning systems are often forced to deal with vague, uncertain or imprecise information. This imprecise nature of information or knowledge is the greatest obstacle to rule induction. Many theories and techniques have been developed in recent years to deal with uncertainty. Among them, the most successful ones are the fuzzy set theory [4] and the Depster–Shafer theory of evidence [5–7]. Rough set theory, which was introduced by Pawlak [8,9] in the early 1980s, provides a novel way of dealing with vagueness and uncertainty. It focuses on the discovery of patterns in incomplete data sets obtained from information source [10,11] and can be used as a basis to perform formal reasoning under uncertainty, machine learning and rules discovery [12–14]. In less than two decades, rough set theory has rapidly established itself in many real-life applications such as medical diagnosis [15], control algorithm acquisition and process control [16] and structural engineering [17]. By coupling rough set theory with genetic algorithms, it is envisaged that an induction engine, which is able to induce probable decision rules from inconsistent information, can possibly be developed.

This paper presents an integrated approach that integrates rough set theory, genetic algorithms (GAs) and Boolean algebra, for rule induction. Using such an approach, a prototype system called RClass-Plus, that discovers rules from inconsistent empirical data, has been developed. The next section, Section 2, reviews the basic notions of rough set theory and GAs. Section 3 briefly describes RClass-Plus. RClass-Plus was validated using the data obtained from a case study. The details of the validation are presented in Section 4. The last section, Section 5, summarises the main conclusions reached in this work.

2. Basic notions

2.1. Rough set theory

The concept of rough set theory overlaps, to a certain extent, with techniques dealing with vagueness and uncertainty such as fuzzy sets and the Depster–Shafer evidence theory. Nevertheless, it can be viewed as an entity of its own right [9]. In particular with fuzzy sets, both of them are neither competing techniques nor the same. Instead, the two techniques complement each other naturally [18–20]. Fuzzy sets theory focuses on the intensity of membership, whereas rough set theory is based on equivalence relations or indiserniblity. Rough set theory is more justified for situations in which the set of empirical or experimental data is too small to employ standard statistical method [9]. Due to the robustness of rough set theory in dealing with uncertainty and vagueness, many researchers attempted to combine it with inductive learning techniques so as to achieve better results. Yasdi [21] combined rough set theory with neural network to deal with learning of imprecise information. Khoo et al. [32] developed a prototype system based on rough set and decision tree learning methodology to perform inductive learning under noisy environment.

Essentially, rough set theory employs a set of multi-valued attributes known as *condition attributes*, and *decision attributes* to classify objects. The information about the objects is represented in a structure known as *information system*, which can be viewed as a *table* with its rows and columns corresponding to the objects and attributes, respectively (Table 1). For example, an information system (S) with 4-tuple can be expressed as follows:

$$S = \langle U, Q, V, \rho \rangle, \tag{1}$$

Table 1
An information system[a]

Objects U	Attributes		Decisions d
	q_1	q_2	
x_1	1	0	0
x_2	1	1	1
x_3	1	2	1
x_4	0	0	0
x_5	0	1	0
x_6	0	2	1
x_7	0	1	1
x_8	0	2	0
x_9	1	0	0
x_{10}	0	0	0

[a] x_i ($i = 1, 2, \ldots, 10$): objects of the set U; q_i ($i = 1$ and 2): condition attributes; d: decision attribute.

where U is the *universe* which contains a finite set of objects, Q is a finite set of attributes, $V = \cup_{q \in Q} V_q$ and V_q is a domain of the attribute q, and $\rho : U \times Q \rightarrow V$ is the information function such that $\rho(x, q) \in V_q$ for every $q \in Q$ and $x \in U$ and any pair (q, v), $q \in Q$, $v \in V_q$ is called a *descriptor* in S.

Imprecise information causes *indiscernability* of objects. This *indiscernibility relation* is an equivalence relation on U. For example, Table 1 shows an information system with inconsistent data. A *conflict* (or *inconsistency*) exists between objects x_5 and x_7 because they are indiscernible by condition attributes q_1 and q_2 and have different decision attributes (d). Similarly, another conflict also exists between x_6 and x_8.

Objects x and y which are indiscernible by a set of attributes P in S, can be expressed as

$$x, y \in U, P \in Q \text{ and } x\hat{P}y,$$
$$\text{iff } \rho(x, q) = \rho(y, q) \text{ for every } q \in P.$$

Equivalence classes of relation $\hat{P}$ are called the P-elementary sets in S. Any finite union of P-elementary sets is known as a *P-definable* set in S; the Q-elementary sets are called the *atoms* in S; and the decision-elementary sets are known as *concepts*.

Rough set theory offers a powerful means to deal with the bespoke inconsistency problems. It uses a set of lower and upper approximations as its main vehicles for problem solving. The upper and lower approximations represent the classes of indiscernible objects that possess sharp descriptions on concepts but with no sharp boundaries. For a given concept, C, the *lower approximation* (denoted as $\underline{R}(C)$) represents a set of objects in U, which can be *certainly* classified as belonging to concept C by a set of attributes, R, viz. *certain data set*, and can be defined as

$$\underline{R}(C) = \cup \left\{ \frac{Y \in U}{R : Y \subseteq C} \right\}. \tag{2}$$

On the other hand, the *upper approximation* (denoted as $\overline{R}(C)$) represents a set of objects in U that can be *possibly* classified as belonging to concept C by R, viz. *possible data set*, such that

$$\underline{R}(C) = \cup \left\{ \frac{Y \in U}{R : Y \cap C \neq \emptyset} \right\}. \tag{3}$$

Elements belonging only to the upper approximation compose the *boundary region* (denoted as $\mathrm{BN_R}$), and can be expressed as

$$\mathrm{BN_R}(C) = \overline{R}(C) - \underline{R}(C). \tag{4}$$

This boundary region represents the set of objects that cannot be certainly classified as belonging to concept C by R. Such a concept C is called a *rough set*. In other words, rough sets are sets having non-empty boundary regions.

The above notions can be applied to the information system depicted in Table 1. The universe, U, consists of 10 objects and can be described using two concepts, concepts '0' and '1' (Fig. 1). As already mentioned, two conflicts, namely objects x_5 and x_7, and objects x_6 and x_8, exist in the data sets. These conflicts cause the objects to be *indiscernible* and constitute the doubtful area which is denoted by $\mathrm{BN_R}(0)$ or $\mathrm{BN_R}(1)$ as shown in Fig. 1. The lower and upper approximations of concept '0' are given by objects '1, 4, 9 and 10' (certain training data set of concept '0') and '1, 4, 5, 6, 7, 8, 9, and 10' (possible training data set of concept '0'), respectively. Concept '1' can be interpreted similarly.

2.2. Genetic algorithms (GAs)

Genetic algorithms are stochastic and evolutionary search techniques based on the principles of biological evolution, natural selection, and genetic recombination. They simulate the principle of 'survival of the fittest' in a population of potential solutions known as *chromosomes*. Each chromosome represents one possible solution to the problem or a rule in a classification. The population evolves over time through a process of competition whereby the fitness of each

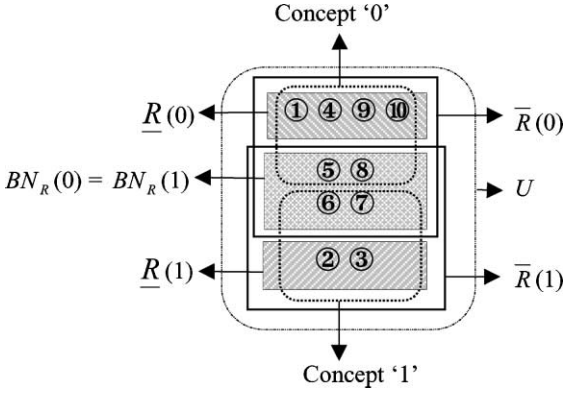


Fig. 1. Basic notions of rough set theory.

of the chromosomes is evaluated using a fitness function. Genetic algorithm is conceptually simple but computationally powerful, which has made it a promising research area since its inception. GAs have been used to handle a wide variety of applications, especially in the optimisation, search and machine learning [3,23–25]. Genetic algorithm-based learning techniques take advantage of the unique search engine of GAs to glean probable decision rules from its search space.

Over the past decades, GAs have evolved from the basic algorithms to various enhanced versions. However, the basic principles used in GAs remain the same. In this work, only the basic notions of GAs are presented. The mechanism of a general model of GA is very simple. The algorithm begins with a population of chromosomes generated either randomly or from some set of known specimens, and cycles through the three steps, namely evaluation, selection, and reproduction. A chromosome contains the information about the solution to a problem, which it represents. Typically, it can be encoded using a binary string as follows:

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

Other than using a binary string, depending on the complexity of a problem domain, a chromosome can also be encoded using a series of integers or floating point variables [26]. Fig. 2 shows a general outline of the algorithm. In the first step, that is, the evaluation step, each string is evaluated according to a given performance criterion known as *fitness function*, and assigned a *fitness score*. The fitness function is designed here in order that the fitness score of an individual or a group of individuals moves toward an extremum as its performance improves. In the next step, the selection step, a decision is made according to the fitness score assigned to each individual to decide which individuals are permitted to produce offspring and with what probability. Finally, the reproduction step involves the creation of offspring chromosomes by two genetic operators, namely cross-over and mutation. This is the most important part of the genetic algorithms as these genetic operators have an impact on the performance of GAs. Typically, the cross-over operator randomly selects a cross-site and swaps the genes of two parent chromosomes to produce two offspring chromosomes. This can be

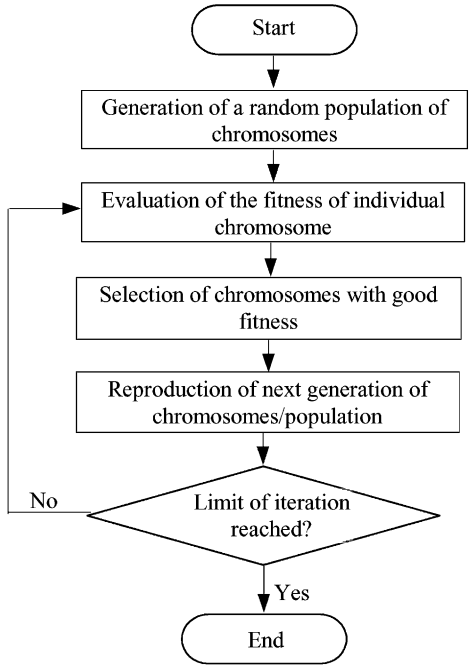


Fig. 2. A typical GA program flow.

easily illustrated using a pair of chromosomes encoded as two binary strings, where the cross-site is denoted by "|" in each chromosome as follows:

Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

Upon the completion of a cross-over operation, mutation takes place. This is to prevent the solution converges prematurely. For a chromosome encoded as a binary string, genes are randomly selected to undergo mutation operation, where '1' is changed into '0' or vice versa, in the following manners:

Original offspring 1	1101111000011110
Mutated offspring 1	1100111000011110
Original offspring 2	1101100100110110
Mutated offspring 2	1101101100110100

The bespoke cycle continues for a predetermined number of *generations*, or until an acceptable performance level is achieved, and the algorithm ends.

3. RClass-Plus

As already mentioned, inductive learning systems are often forced to deal with vague, uncertain or imprecise information in reality. Many inductive learning systems such as ID3, ID4 and ID5, are not capable of handling imprecise information or training data set effectively [27,28]. Based on rough set theory, Grzymala-Busse developed a system known as LERS for inductive learning [29,30]. LERS is able to deal with the inconsistencies in the training data. However, as observed by Grzymala-Busse [30], LERS becomes impractical when the training data set is huge. This can possibly be attributed to the complexity of its computational algorithm. Furthermore, the rules induced by LERS are relatively complicated and difficult to understand or interpret as compared to those generated by other inductive learning systems. RClass-Plus adopts an approach that integrates rough set theory, genetic algorithms and Boolean algebra. It has:

- the ability of handling inconsistent training data which is achieved by the incorporation of rough set theory;
- a simple but effective GA-based search engine to induce probable decision rules; and
- a Boolean algebra-based mechanism to prune and simplify the probable decision rules induced by GAs.

The framework of RClass-Plus is depicted in Fig. 3. RClass-Plus comprises four modules, namely a pre-processor, a rough set analyser, a GA-based search engine, and a rule pruner. The knowledge gleaned from the process or experts is stored and, subsequently forwarded to the prototype system for classification and rule induction (Fig. 4).

The pre-processor module reads in the input data file and automatically initialises all the parameters necessary for the GA-based search engine such as the length of chromosomes, the population size, the number of generation, and the probabilities of cross-over and mutation.

The rough set analyser performs two tasks: the consistency check on the input data set and the classification of objects based on rough set theory. For an inconsistent data set, three sub-tasks, namely the identification of attributes, concept forming and approximation, are performed.

The approximation sub-task involves using the lower and upper approximators developed in this work to estimate the lower and upper approximations in accordance to the rough set theory. During which, the input training data set is split into *certain training data set* and *possible training data set* by the lower approximator and upper approximator, respectively. Subsequently, these training data sets, that is, the certain training data set and the possible training data set, are forwarded to the GA-based search engine for rule extraction. As for the consistent training data set, it is saved as a certain training data set and routed to the GA-based search engine.

The genetic-based search engine, once invoked, performs the bespoke genetic operations such as reproduction, cross-over and mutation, to extract *certain rules* and *possible rules* from the *certain training data set* and *possible training data set*, respectively.

The rule pruner performs two tasks: pruning and simplifying. It examines the rules, both *certain* and *possible* rules, extracted by the GA-based search

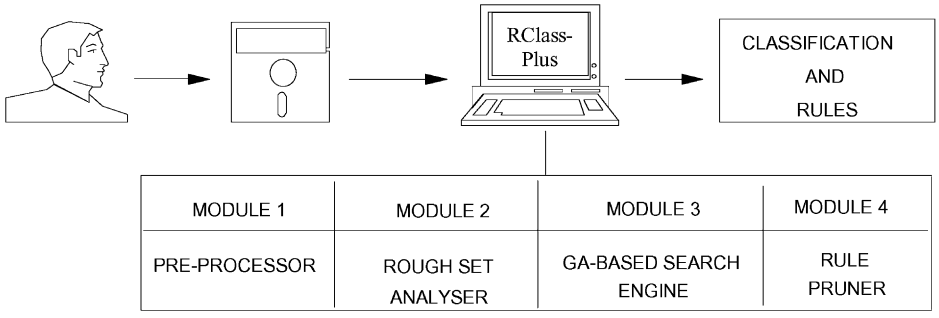


Fig. 3. Framework of RClass-Plus.

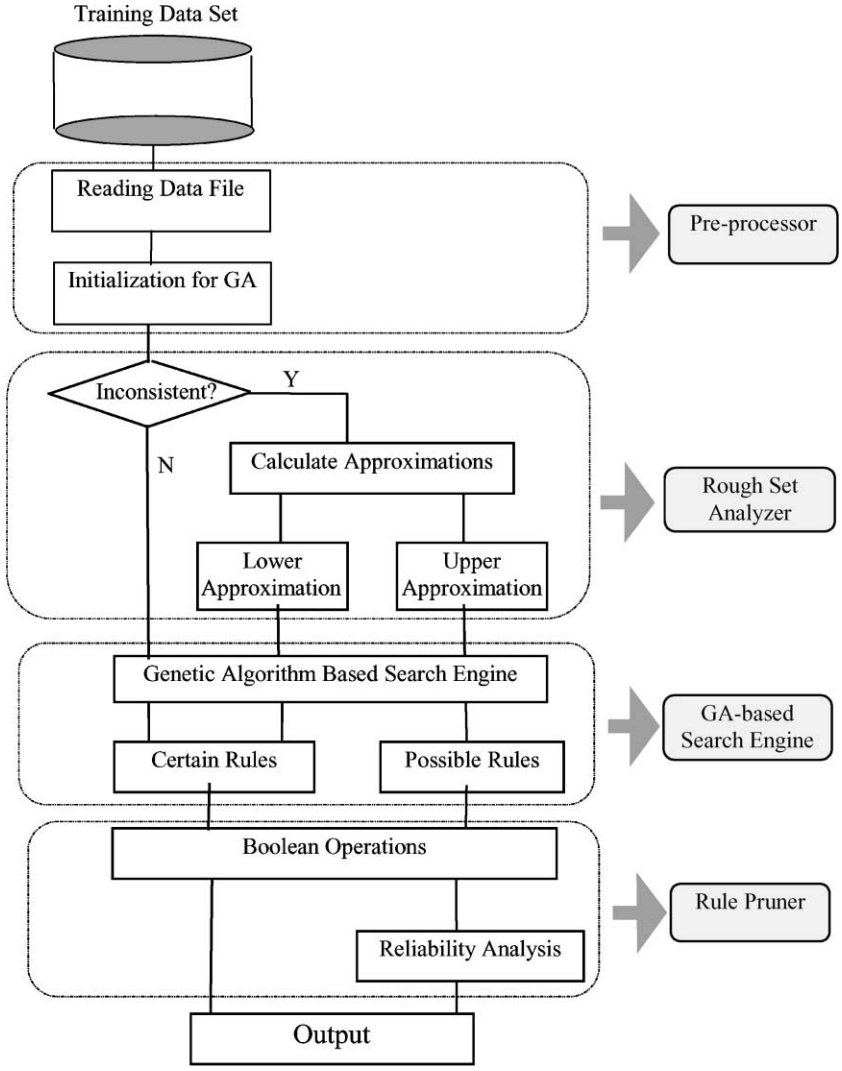


Fig. 4. Details of RClass-Plus.

engine and uses Boolean operators such as union and intersection, to prune and simplify the rules. During the pruning operation, redundant rules are removed. Whereas, related rules are clustered and generalised during simplification. As *possible* rules are not definitely *certain*, the quality and reliability of these *possible* rules must therefore be assessed. For every *possible* rule, the prototype system also estimates its reliability using an index, which is defined as the ratio of the number of observations that are correctly classified by a *possible* rule, and the number of observations whose condition attributes are covered by the same rule in the original training data set as follows:

$$\text{reliability index} = \frac{\text{Observation\_Possible\_Rule}}{\text{Observation\_Possible\_Original\_Data}}, \quad (5)$$

where Observation_Possible_Rule is the number of observations that are correctly classified by a possible rule and Observation_Possible_Original_Data is the number of observations with condition attributes covered by the same rule in the original data set.

This index can therefore be viewed as the probability of classifying the inconsistent training data set correctly. For each *certain* rule extracted from the *certain* training data set, a so-called completeness index which is defined as the ratio of the number of observations that are correctly classified by a *certain* rule, and the number of observations whose condition attributes are covered by the same rule in the original training data, is also calculated as follows:

$$\text{completeness index} = \frac{\text{Observation\_Certain\_Rule}}{\text{Observation\_Certain\_Original\_Data}}, \quad (6)$$

where Observation_Certain_Rule is the number of observations that are correctly classified by a certain rule and Observation_Certain_Original_Data is the number of observations with condition attributes covered by the same rule in the original training data.

Such an index indicates the amount of observations in the original training data set that are related to the *certain* rule. In other words, it represents the usefulness or the effectiveness of the *certain* rule. The reliability and completeness indices are included as part of the system's output and are displayed in the parentheses following the rules induced.

4. A case study

RClass-Plus was validated using the data obtained from a case study, which monitors the condition of a machine. Preliminary observations show that the condition of the machine is related to its cooling water temperature (attribute CWTemp), the noise level (attribute Noise), and the level of vibration (attribute Vib). The results of a series of on-site observations are recorded and summarised in Table 2. For simplicity, two states namely normal and high, are used to describe the condition of attribute CWTemp. For attribute Noise, its condition is described by two different states: low and high. In the case of attribute Vib, three states, namely normal, violent, extremely-violent, are used. The condition of the machine can be classified into two states: normal and faulty. In order to process the information, the *linguistic descriptions* of the conditions (for the attributes and the machine's state) need to be transformed into real numbers. This is done by using the following transformation scheme:

normal $\Rightarrow$ 0; low $\Rightarrow$ 0;
faulty $\Rightarrow$ 1; high $\Rightarrow$ 1;
violent $\Rightarrow$ 1; extremely-violent $\Rightarrow$ 2.

The transformed results are depicted in Table 3. Clearly, the states of observations 5 and 7 contradict one another.

As the training data set contains conflicting observations, the rough set analyser proceeds to identify the attributes, perform concept forming and carry out approximation. Two concepts, namely C_1 (state = normal) and C_2 (state = faulty) are identified.

Table 2
Machine condition and its parameters

Observation	CWTemp	Noise	Vib[a]	State
1	High	Low	Normal	Normal
2	High	Low	Violent	Faulty
3	High	High	Extremely-violent	Faulty
4	Normal	High	Normal	Normal
5	Normal	Low	Violent	Normal
6	Normal	High	Extremely-violent	Faulty
7	Normal	Low	Violent	Faulty
8	Normal	Low	Extremely-violent	Normal
9	High	High	Normal	Normal
10	Normal	Low	Normal	Normal
11	High	High	Violent	Faulty
12	High	Low	Extremely-violent	Faulty
13	Normal	High	Violent	Faulty

[a] Vib: vibration.

Table 3
Transformed machine condition and its parameters

Observation	CWTemp	Noise	Vib[a]	State
1	1	0	0	0
2	1	0	1	1
3	1	1	2	1
4	0	1	0	0
5	0	0	1	0
6	0	1	2	1
7	0	0	1	1
8	0	0	2	0
9	1	1	0	0
10	0	0	0	0
11	1	1	1	1
12	1	0	2	1
13	0	1	1	1

[a] Vib: vibration.

The lower and upper approximations of these concepts are then calculated. Let x_i ($i = 1, \ldots, 13$) denote the observation numbers as shown in Tables 2 and 3. More specifically, for concept $C_1 = \{x_1, x_4, x_5, x_8, x_9, x_{10}\}$, the lower and upper approximations are, respectively given by

$$\underline{R}(C_1) = \{x_1, x_4, x_8, x_9, x_{10}\} \quad \text{and}$$
$$\overline{R}(C_1) = \{x_1, x_4, x_5, x_7, x_8, x_9, x_{10}\}.$$

Thus, the boundary region of concept C_1 is given by

$$\mathrm{BN_R}(C_1) = \overline{R}(C_1) - \underline{R}(C_1) = \{x_5, x_7\}.$$

Similarly, for concept $C_2 = \{x_2, x_3, x_6, x_7, x_{11}, x_{12}, x_{13}\}$, the lower and upper approximations and the boundary region are given by

$$\underline{R}(C_2) = \{x_2, x_3, x_6, x_{11}, x_{12}, x_{13}\};$$
$$\overline{R}(C_2) = \{x_2, x_3, x_5, x_6, x_7, x_{11}, x_{12}, x_{13}\};$$
$$\mathrm{BN_R}(C_2) = \overline{R}(C_2) - \underline{R}(C_2) = \{x_5, x_7\}.$$

The GA-based search engine is then used to extract rules from the *certain* and *possible* training data sets obtained by the rough set analyser. It randomly generates 30 chromosomes to form an initial population of possible solutions, that is, the chromosomes. These chromosomes are coded using the scheme depicted in Table 4.

For chromosome representation and the corresponding operations, RClass-Plus adopts the traditional binary string representation and its corresponding cross-over and mutation operators. Using this scheme, each chromosome is expressed as a binary string, that is, a string comprises '0' and '1'.

Using such a scheme, a classification rule can be easily represented by an 8-bit chromosome. For instance, the rule "if (CWTemp $\geq$ high) and (Noise $\geq$ low) and (Vib $\geq$ violent) then the state of the machine is faulty" can be coded as 11101011. Apparently, such a representation is rather effective in performing the cross-over and mutation operations.

Other than choosing a good scheme for chromosome representation, it is important to define a reasonable fitness function that rewards the right kind of chromosomes. Basically, the purpose of applying GAs in this research is to extract rules that can maximise the probability of classifying the objects correctly. Thus, the fitness value of a chromosome can be described by its reliability, or in other words, the probability to classify objects in a training data set correctly. Mathematically, the fitness function used in this work is expressed as

$$\text{fitness of a chromosome} = \left(\frac{\text{number of objects classified correctly by the rule}}{\text{number of objects covered by the condition of the rule}}\right)^2. \tag{7}$$

Table 4
Chromosome coding for machine condition

Bit 1, 3 and 5: operator	0 = Less than or equal to ($\leq$) 1 = Greater than or equal to ($\geq$)
Bit 2: cooling water temperature (CWTemp)	0 = Normal 1 = High
Bit 4: Noise	0 = Low 1 = High
Bit 6 and 7: Vib[a]	00 = Normal 01 = Violent 10 = Extremely-violent
Bit 8: state of machine	0 = Normal 1= Faulty

[a] '11' stands for no operation.

For example, if a rule (represented by a chromosome) can correctly classify five objects in a training data set, and if there are six objects having the same condition attribute–value pairs as the said rule, then the fitness value of this chromosome is $(5/6)^2 = 0.6944$. The square operator appeared in the fitness function is to ensure rapid convergence. It is used to suppress bad chromosomes with low fitness scores and promote the creation of good chromosomes with high fitness scores. Thus, the above fitness function favours rules that can classify objects correctly. Furthermore, it also satisfies both consistency and completeness criteria, which are of great importance for the evaluation of a rule. A rule is said to be consistent if it covers no negative sample, that is,

no object in the training data set violating the rule; and it is said to be complete if the rule is able to cover all the positive samples that satisfy the condition of the rule in the training data set [31]. As previously mentioned, after evaluating the fitness values of chromosomes, those chromosomes with above average cfitness values are selected for reproduction. As for cross-over and mutation, the respective probabilities are fixed at 0.85 and 0.01. With a higher probability of cross-over, offspring chromosomes that maintain the genetic traits of the parent chromosomes can be generated easily. This allows chromosomes with higher fitness values, that is, better solutions, to be discovered. A lower probability of mutation prevents the search for optimal solutions to degenerate into a random one.

The rule set induced by the GA-based search engine may contain rules with identical fitness values. Some of these rules can be combined to form a more general or concise rule using Boolean algebra. The rule pruner is assigned to detect and solve the redundancy problem. As an simple example, consider the following rules:

Rule 1 :

if CWTemp $\geq$ 1 and Vib $\geq$ 1 then state = faulty,

Rule 2 :

if CWTemp $\geq$ 0 and Vib $\geq$ 1 then state = faulty.

Obviously, from the Boolean logic perspective, rule 1 $\subset$ rule 2. The rule pruner will then proceed to combine the rules and produce the following rule:

if CWTemp $\geq$ 0 and Vib $\geq$ 1 then state = faulty.

The results generated by RClass-Plus are shown in Table 5. Two sets of rules are available. As already mentioned, the value recorded in the parentheses following each *certain* and *possible rule* represents the completeness and the reliability indices, respectively. All the indices are represented in fraction form, with the numerator corresponding to the number of correctly classified observations and the denominator corresponding to the number of observations whose condition attributes are covered by the rule. Analysis shows that the rules induced by RClass-Plus are simple, reasonable and logical.

Table 5
Rules induced by the RClass-Plus[a]

Rules generated by RClass-Plus
Certain rules
IF {Vib $\leq$ normal} THEN State = normal (4/4)
IF {CWTemp $\geq$ high} & {Vib $\geq$ violent} THEN State = faulty (4/4)
IF {Noise $\geq$ high} & {Vib $\geq$ violent} THEN State = faulty (4/4)
Possible rules
IF {CWTemp $\leq$ normal & Noise $\leq$ low} THEN State = normal (3/4)
IF {Noise $\leq$ low & Vib $\leq$ violent} THEN State = normal (3/5)
IF {CWTemp $\leq$ normal & Vib $\leq$ violent} THEN State = normal (3/5)
IF {Vib $\geq$ violent} THEN State = faulty (7/9)
IF {CWTemp $\geq$ high} THEN State = faulty (4/6)
IF {Noise $\geq$ high} THEN State = faulty (4/6)

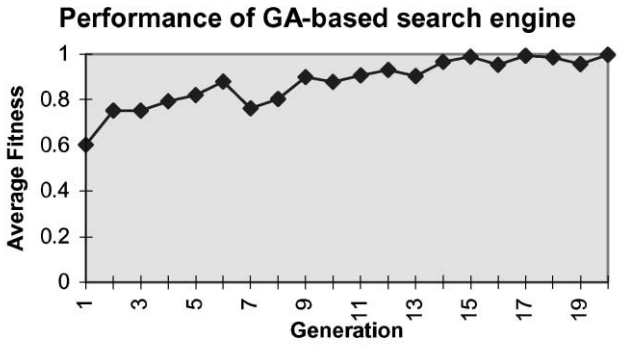
[a] The same input data set was tested on ID3. With the inconsistent data, the ID3 system hung. However, by removing the inconsistent incidents, ID3 was able to produce rules identical to the certain rules induced by the RClass-Plus.

5. Discussion

The above example shows that RClass-Plus is able to support the twin-concept classification of objects proposed by Pawlak [9]. RClass-Plus has successfully integrated the basic notions of rough set theory with the GA-based search engine and Boolean algebra to yield a new approach for inductive learning under uncertainty. The entropy-based induction engine used in RClass [32] has been replaced by the GA-based search engine. Through this integration, the proposed system has combined the strengths of rough set theory and the GA-based search mechanism. RClass-Plus was compared with other inductive learning techniques. The results of the comparison are shown in Table 6. By removing the inconsistency in the training data set, only *certain* rules are generated. These *certain* rules are identical to those produced by ID3. With the help of Boolean algebra, the rules induced by the prototype system are simple and concise as compared to those produced by LERS. Compared to its predecessor, RClass, more logical rules have also been generated. The GA-based engine is efficient. In the case study, it converges within 20 generations for both *possible* and *certain* training data sets (Fig. 5).

Table 6
Capabilities of ID3, LERS, RClass, and RClass-Plus

Technique	Dealing with uncertainty and inconsistency	Simple and concise rules induction	Extracting complete rules
ID3	No	Yes	For consistent data set only
LERS	Yes	No	Not evaluated
RClass	Yes	Yes	No
RClass-Plus	Yes	Yes	Yes



Note: population size = 30; crossover rate = 0.85; mutation rate = 0.01;

Fig. 5. Performance of the GA-based search engine.

The ability to induce simple and concise rules has the following advantages:

- Easy to understand.
- Easy to interpret and analyse.
- Easy to validate and cross-check.

The ability to handle uncertainty or inconsistent information that is common in reality makes RClass-Plus more versatile in real industrial applications. Manufacturing diagnosis, for example, is an important area closely linked to productivity. Frequently, it requires diagnostic rules to be induced from examples or empirical data obtained from the manufacturing system. As modern manufacturing systems become larger and more complicated, due to interactions, the relationships among the various sub-systems or system components can no longer be identified easily. Furthermore, the relationships may be blurred by undesirable inputs from the environment and make them less obvious to the engineers. A rule induction system such as RClass-Plus that is able to handle imprecision is therefore needed. Once the diagnostic rules about the system have been established, a near real-time diagnostic system based on the work in [22] can be easily developed.

6. Conclusions

This work has successfully shown that it is possible to integrate rough set theory with a GA-based search algorithm to perform rule extraction from examples with inconsistent data. Using this novel approach, a prototype rule induction system called RClass-Plus, has been developed. It has shown that RClass-Plus is able to combine the strengths of rough set theory and

the GA-based search algorithm to deal with rule induction under uncertainty. Two sets of rules, certain rules and possible rules, can be induced from examples by RClass-Plus. The data extracted from a case study was used to validate the proposed system. Analysis shows that the rules both logical and complete. By removing the inconsistency in the training data sets, only *certain* rules are generated. These rules are identical to those of ID3. With the help of Boolean algebra, the rules produced are simple and concise as compared to those produced by LERS. Compared its predecessor, RClass, more logical rules have been generated. For all the rules generated, RClass-Plus is also able to provide an estimation of the expected reliability in fraction form. These would assist the users in assessing the appropriateness of the rules generated.

References

[1] S.K.M. Wong, W. Ziarko, Y.R. Li, Comparison of rough-set and statistical methods in inductive learning, International Journal of Man-Machine Studies 24 (1986) 53–72.

[2] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, 1992.

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

[4] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.

[5] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, W. Ziarko, Rough sets, Communications of the ACM 38 (11) (1995) 89–95.

[6] Z. Pawlak, Rough sets and fuzzy sets, Fuzzy Sets and Systems 17 (1985) 99–102.

[7] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, 1976.

[8] Z. Pawlak, Rough sets, International Journal of Computer and Information Sciences 11 (5) (1982) 341–356.

[9] Z. Pawlak, Rough Sets — Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.

[10] R. Slowinski, J. Stefanowski, Rough classification in incomplete information systems, Mathematical and Computer Modeling 12 (10/11) (1989) 1347–1357.

[11] Z. Pawlak, Why rough sets, 1996 IEEE International Conference on Fuzzy Systems 2 (1996) 738–743.

[12] W. Ziarko, Rough sets and knowledge discovery: an overview, in: Proceedings of the International Workshop on Rough Sets and Knowledge Discovery, Rough Sets, Fuzzy Sets and Knowledge Discovery, 1994, pp. 11–15.

[13] Z. Pawlak, Rough classification, International Journal of Man-Machine Studies 20 (1984) 469–483.

[14] Y.Y. Yao, S.K.M. Wong, T.Y. Lin, A review of rough sets models, Rough Sets and Data Mining — Analysis for Imprecise Data, 1997, pp. 47–76.

[15] K. Slowinski, Rough classification of HSV patients, Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, 1992, pp. 77–94.

[16] A. Mrozek, Rough sets in computer implementation of rule-based control of industrial process, Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, 1992, pp. 19–32.

[17] T. Arciszewski, W. Ziarko, Inductive learning in civil engineering: a rough sets approach, Microcomputers in Civil Engineering 5 (1) (1990) 19–28.

[18] D. Dubois, H. Prade, Rough fuzzy sets and fuzzy rough sets, International Journal of General Systems 17 (1990) 191– 209.

[19] T.Y. Lin, Fuzzy reasoning and rough sets, in: Proceedings of the International Workshop on Rough Sets and Knowledge Discovery, Rough Sets, Fuzzy Sets and Knowledge Discovery, 1994, pp. 343–348.

[20] D. Dubois, H. Prade, Putting rough sets and fuzzy sets together, Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, 1992, pp. 203–231.

[21] R. Yasdi, Combining rough sets learning- and neural learning-method to deal with uncertain and imprecise information, Neurocomputin 7 (1995) 61–84.

[22] L.P. Khoo, C.L. Ang, J. Zhang, A graph theoretic approach with fuzzy critique to manufacturing diagnosis, Integrated Product and Process Development, 1998, pp. 265–278.

[23] L. Davis (Ed.), Handbook of Genetic Algorithms, Norstrand Reinhold, New York, 1991.

[24] F. Stephanie (Ed.), Proceedings of the fifth International Conference on Genetic Algorithms, University of Illinois at Urbana, Champaign, 1993.

[25] J.J. Grefenstette (Ed.), Genetic Algorithms for Machine Learning, Kluwer Academic Publishers, Boston, 1994.

[26] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, New York, 1992.

[27] D.T. Pham, M.S. Aksoy, A new algorithm for inductive learning, Journal of Systems Engineering 5 (1995) 115–122.

[28] D.T. Pham, S.S. Dimov, An efficient algorithm for automatic knowledge acquisition, Pattern Recognition 30 (7) (1997) 1137–1143.

[29] J.W. Grzymala-Busse, C.P. Wang, Classification and rule induction based on rough sets, 1996 IEEE International Conference on Fuzzy Systems 2 (1996) 744–747.

[30] J.W. Grzymala-Busse, LERS — a system for learning from examples based on rough sets, Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, 1992, pp. 3–8.

[31] K.A. De Jong, W.M. Spears, D.F. Gordon, Using genetic algorithms for concept learning, Machine Learning 13 (1993) 161–188.

[32] L.P. Khoo, S.B. Tor, L.Y. Zhai, A rough-set-based approach for classification and rule induction, International Journal of Advanced Manufacturing 15 (1999) 438–444.

Li-Pheng Khoo is an Associate Professor in the School of Mechanical and Production Engineering, Nanyang Technological University, Singapore. He graduated from the University of Tokyo in 1978 and earned his MSc in Industrial Engineering from the National University of Singapore and his PhD from the University of Wales in the UK. His research interests are in artificial intelligence, concurrent engineering and design for manufacture and assembly. He is currently the Head of Manufacturing Engineering Division.

Lian-Yin Zhai received the BEng degree from Xi'an Jiaotong University and the MEng degree from Nanyang Technological University. He is currently a Research Associate at the School of Mechanical and Production Engineering, Nanyang Technological University, Singapore.