PERGAMON

Expert Systems with Applications 19 (2000) 9–17

Expert Systems with Applications

www.elsevier.com/locate/eswa

A coverage-based genetic knowledge-integration strategy

Ching-Hung Wang[a], Tzung-Pei Hong[b,*], Ming-Bao Chang[c], Shian-Shyong Tseng[c]

[a]*Chunghwa Telecommunication Laboratories, Chung-Li 32617, Taiwan, ROC*
[b]*Department of Information Management, I-Shou University, Kaohsiung 84008, Taiwan, ROC*
[c]*Institute of Computer and Information Science, National Chiao-Tung University Hsin-Chu, 30050, Taiwan, ROC*

Abstract

In this paper, we propose a coverage-based genetic knowledge-integration approach to effectively integrate multiple rule sets into a centralized knowledge base. The proposed approach consists of two phases: knowledge encoding and knowledge integration. In the knowledge-encoding phase, each rule in the various rule sets that are derived from different sources (such as expert knowledge or existing knowledge bases) is first translated and encoded as a fixed-length bit string. The bit strings combined together thus form an initial *knowledge population*. In the knowledge-integration phase, a genetic algorithm applies genetic operations and credit assignment at each rule-string to generate an optimal or nearly optimal rule set. Experiments on diagnosing brain tumors were made to compare the accuracy of a rule set generated by the proposed approach with that of the initial rule sets derived from different groups of experts or induced by various machine learning techniques. Results show that the rule set derived by the proposed approach is more accurate than each initial rule set on its own.
© 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Brain tumor diagnosis; Genetic algorithm; Knowledge encoding; Knowledge integration; Credit assignment

1. Introduction

Developing an expert system requires construction of a complete, consistent, and unambiguous knowledge base. Conventional approaches to knowledge base construction involve interaction and dialogue between knowledge engineers and domain experts. Gaines and Shaw mentioned that the knowledge required to develop knowledge-based systems is often distributed among groups of experts rather than being available for elicitation from a single expert (Gaines & Shaw, 1993). Acquiring and integrating multiple knowledge inputs from many experts or by various knowledge-acquisition techniques thus plays an important role in building effective knowledge-based systems (Baral, Kraus & Minker, 1991; Boose, 1985; Boose & Bardshaw, 1987; Gaines, 1989). Integrating multiple knowledge sources in developing an expert system has the following benefits (Medsker, Tan & Turban, 1995):

1. Existing knowledge can be reused.
2. Knowledge acquired from different sources usually has better validity than that from only one source.
3. The resulting knowledge base is usually more comprehensive than that from only one source.
4. Knowledge integration by computers facilitates faster and more inexpensive building of knowledge-based systems than that by human experts.

Generally, knowledge integration can be though of as a multi-objective optimization problem (Yuan & Zhuang, 1996), in which, due to the large search space, it is very difficult to find an optimal solution. A genetic algorithm (GA) was usually used to discover a desirable but not necessarily optimal set of rules (Cordon, Herrera & Lozano, 1997; Holland, 1985; Holland & Reitman, 1983; Smith, 1980). Several knowledge-integration strategies based on genetic search have been proposed (Wang, Hong & Tseng, 1997; Wang, Hong & Tseng, 1998a; Wang, Hong, Tseng & Liao, 1998b; Wang, Hong & Tseng, 2000). Wang et al. proposed GA-based knowledge integration strategies that operated at the rule-set level to automatically integrate multiple rule sets in a distributed-knowledge environment (Wang et al., 1997, 1998b, 2000). In addition, a self-integrating knowledge-based brain tumor diagnostic system based on these strategies was successfully developed (Wang, Hong & Tseng, 1996).

In this paper, we propose a coverage-based genetic knowledge-integration approach which operates at the rule

* Corresponding author.
E-mail addresses: ching@ms.chttl.com.tw (C.-H. Wang), tphong@isu.edu.tw (T.-P. Hong), changbao@ms7.hinet.net (M.-B. Chang), sstseng@cis.nctu.edu.tw (S.-S. Tseng).

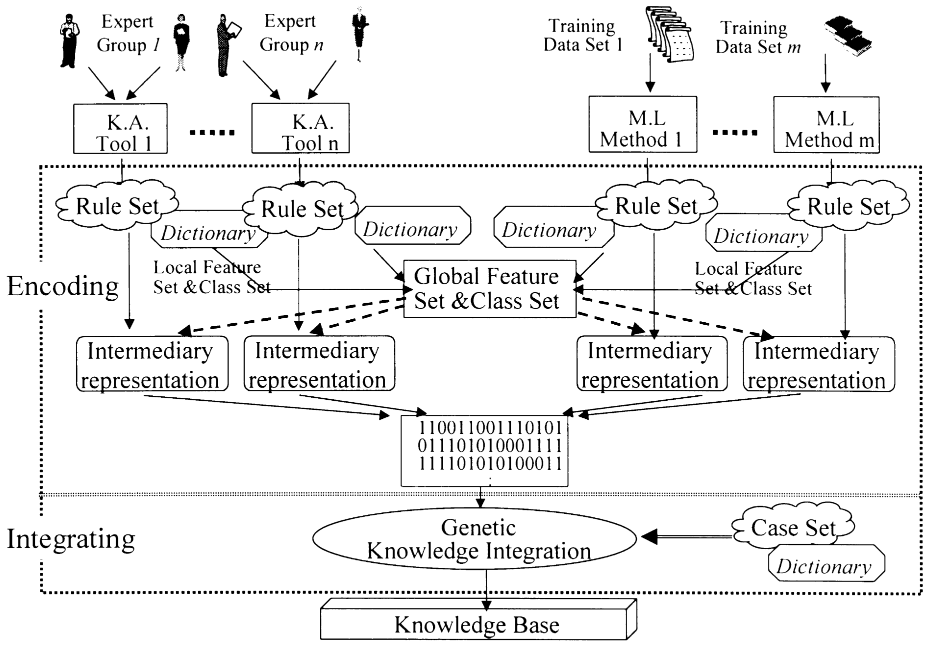

Fig. 1. A genetic knowledge-integration framework.

level for effectively integrating multiple rule sets into one centralized knowledge base. The proposed approach takes less memory and processing time than those in (Wang et al., 1998b). It does not need to apply any domain-specific genetic operators to solve the redundancy (Giarratano & Riley, 1993), subsumption (Giarratano & Riley, 1993), and contradiction (Giarratano & Riley, 1993) problems. Instead, it used an instance-coverage approach to effectively solve them. In addition, domain experts need not intervene in the integration process since the work is executed by computers.

The use of this approach to develop a brain tumor diagnostic system is also described. Results show that the knowledge base derived by our approach is much more accurate than each initial rule set on its own. These initial rule set sets may be acquired from different experts or induced by various machine learning techniques. Results also show that knowledge integration is a successful application of genetic classifier systems.

The remainder of this paper is organized as follows. A genetic knowledge-integration framework is proposed in Section 2. The knowledge-encoding strategy used in the proposed approach is stated in Section 3. The knowledge-integration strategy used in the proposed approach is stated in Section 4. Experimental results on the brain tumor diagnosis are reported in Section 5. Conclusions and future work are given in Section 6.

2. A genetic knowledge-integration framework

The proposed genetic knowledge-integration framework is shown in Fig. 1. Four types of knowledge and data, including *knowledge sets*, *knowledge dictionaries*, *data sets* and *data dictionaries*, may be obtained from various sources. Knowledge from each source might be directly obtained by a group of human experts using a knowledge-acquisition tool, or derived from a machine-learning method. Here, we assume that all knowledge sources are represented by rules since almost all knowledge derived by knowledge-acquisition (K.A.) tools or induced by machine-learning (M.L.) methods may easily be translated into or represented by rules.

In Fig. 1, each knowledge set (rule set) is associated with one knowledge dictionary; and each data set is associated with one data dictionary. The knowledge dictionary defines the vocabulary set used in the knowledge set. The vocabulary set defines features and classes occurring in the condition and conclusion parts of rule sets. The data set contains a variety of sources including documentary evidence, instances or historical records. The data dictionary defines the vocabulary set used in the data set. The vocabulary set defines features and classes used in the data sets.

The proposed approach consists of two phases: *encoding* and *integration*. The encoding phase first transforms each rule among a rule set into an intermediary representation, and further encodes each intermediary representation as a bit-string structure. After encoding, the bit strings derived from various rule sets are gathered together to form an initial *knowledge population*, which is then ready for integrating. The integration phase chooses bit-string rules for “mating”, gradually creating better offspring rules. The offspring rules then undergo recursive “evolution” until an optimal or nearly optimal set of rules is found. After evolution, all the rules in a population are then combined to form

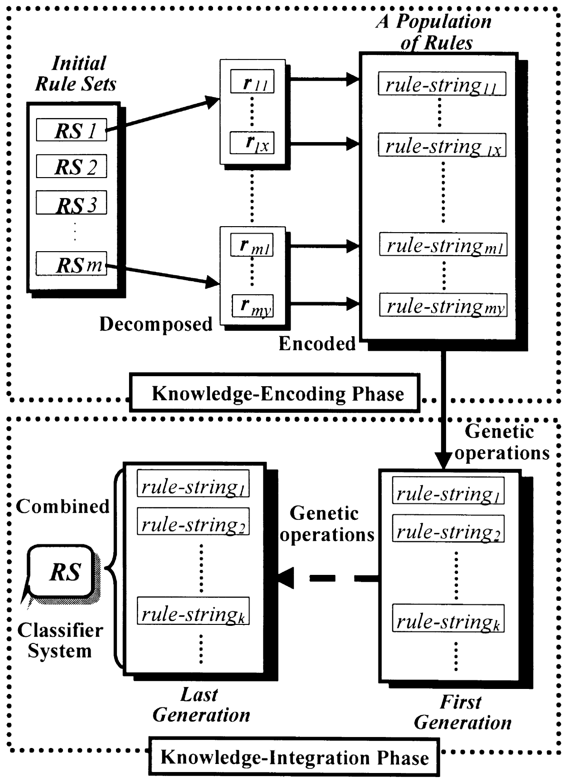

Fig. 2. The illustration of the genetic knowledge-integration algorithm.

a resulting rule set. The proposed genetic knowledge-integration algorithm is stated as follows.

Coverage-based genetic knowledge-integration algorithm:

INPUT: m rule sets from different knowledge sources and a set of test objects including documentary evidence, instances or historical records.
OUTPUT: one integrated rule set.

Knowledge-encoding phase:

STEP 1: collect multiple rule sets from various sources such as expert knowledge or existing knowledge bases.
STEP 2: transform each rule among various sources into an intermediary representation.
STEP 3: encode the intermediary representation as a bit string that will act as an individual in the initial population.

Knowledge-integration phase:

STEP 1: evaluate the fitness value of each rule using the predefined evaluation procedure and the set of test objects.
STEP 2: if the predefined number of generations has been reached, then go to STEP 4; otherwise, do the next step.
STEP 3: select "good" rules and perform *crossover* and *mutation* operations to generate offspring rules; go to STEP 1.
STEP 4: combine all the rules in the population to form a resulting rule set as the final knowledge base.

The proposed algorithm is illustrated in Fig. 2, where RS_1, RS_2, …, RS_m are the rule sets obtained from different knowledge sources, r_{ij} is the j-th rule in the rule set RS_i, and RS is a resulting rule set integrated from RS_1, RS_2, …, RS_m. These two phases are described in detail in the following sections.

3. Knowledge-encoding phase

Having an appropriate data structure to encode rule sets is very important to the genetic learning systems. Two general approaches have been proposed to represent knowledge structures and work on genetic learning systems. One is the classifier systems with genetic operations and credit assignment applied at the rule level, that encode individual rules into fixed-length bit strings (Holland, 1985; Holland & Reitman, 1983; Wilson, 1994). The other is the classifier systems with genetic operations and credit assignment applied at the rule-set level, that encode rule sets into variable-length bit strings (Carse, Fogarty & Munro, 1996; DeJong, Spears & Gordon, 1993). Since each genetic operation in our proposed approach is applied at the rule level, representation of fixed-length rules is preferred here. Each rule is then encoded as a fixed-length chromosome. During encoding, each rule must first be translated into a uniform intermediate representation, which is similar to that in Wang et al. (1998b), to preserve its syntactic and semantic constraints.

The steps for translating each rule into an intermediate representation are described below.

1. Collect the features and possible values occurring in the condition parts of all the rule sets and objects. All features gathered together comprise the global feature set.
2. Collect classes (i.e. possible conclusions) of objects occurring in the conclusion parts of all the rule sets and objects. All classes gathered together comprise the global class set.
3. Translate each rule into an intermediate representation that retains its essential syntax and semantics. If some features in the global feature set are not used by the rule, *dummy* tests are inserted into the condition part of the rule. Each intermediate rule is then composed of N *feature tests* and one *class pattern*, where N is the number of global features collected.

Here, an example for deciding what sport to play according to Sunday's weather (Yuan & Shaw, 1995) is given to demonstrate the process of forming intermediate representation. Three sports {*Volleyball, Swimming, Weight-lifting*} are to be decided by four features {*Outlook, Temperature, Humidity, Wind*}. Assume Feature *Outlook* has three

	Outlook	***Temperature***	***Humidity***	***Wind***	***Sports***
r'_{q1}	111	010	11	01	100

Fig. 3. Bit-string representation of r'_{q1}.

possible values {*Sunny, Cloudy, Rain*}, Feature *Temperature* has three possible values {*Cool, Mild, Hot*}, Feature *Humidity* has two possible values {*Humid, Normal*}, and Feature *Wind* has two possible values {*Windy, Not-windy*}. Also, assume that a rule set RS_q from a knowledge source has the following three rules:

r_{q1}: If (*Temperature* is *Mild*) and (*Wind* is *Not-windy*) then *Volleyball*;
r_{q2}: If (*Outlook* is *Sunny*) and (*Temperature* is *Hot*) then *Swimming*;
r_{q3}: If (*Outlook* is *Rain*) and (*Temperature* is *Hot*) then *Weight-lifting*.

The intermediate representation of these rules would then be:

r'_{q1}: If (*Outlook* is *Sunny* or *Cloudy* or *Rain*) and (*Temperature* is *Mild*) and (*Humidity* is *Humid* or Normal) and (*Wind* is *Not-windy*) then *Volleyball*;
r'_{q2}: If (*Outlook* is *Sunny*) and (*Temperature* is *Hot*) and (*Humidity* is *Humid* or *Normal*) and (*Wind* is *Windy* or *Not-windy*) then *Swimming*;
r'_{q3}: If (*Outlook* is *Rain*) and (*Temperature* is *Hot*) and (*Humidity* is *Humid* or *Normal*) and (*Wind* is *Windy* or *Not-windy*) then *Weight-lifting*.

The tests with underlines are *dummy tests*. Also, r'_{qi} is logically equivalent to r_{qi}, for $i = 1, 2, 3$. After translation, the intermediate representation of each rule is composed of four feature tests and one class pattern.

Although the intermediate representation may include irrelevant tests and increase search space during integration, it can easily map each intermediate rule into a fixed-length string representation. The condition part of each intermediate rule is represented in a conjunctive form with internal disjunctions that can describe complex rules. Irrelevant tests (*don't-cares*) can also be removed in the knowledge decoding process after the knowledge-integration phase is finished.

After translation, each intermediate rule in a rule set is ready for being encoded as a bit string. If the *feature tests* or *class patterns* are numerical, they are first discretized into a number of possible intervals. Each feature test is then encoded into a fixed-length binary string, whose length is equal to the number of possible test values. Each bit thus represents a possible value. For example, the set of legal values for feature *Outlook* is {*Sunny*, *Cloudy*, *Rain*}, and three bits are used to represent this feature. Thus, the bit string *101* would represent the test for *Outlook* being "*Sunny*" or "*Rain*". Similarly, the class pattern is encoded into a fixed-length binary string with each bit representing a possible class. *N* feature tests and one class pattern are then encoded and concatenated together as a fixed-length rule substring. Each rule in the intermediary representation is then encoded as a fixed-length bit string. The length of each rule would then be:

$$Length(rule) = \left[\sum_{i=1}^{N} number\ of\ possible\ values\ for\ Feature(i)\right] + number\ of\ Classes,$$

where *N* is the number of features. The process of encoding intermediate rules is illustrated by the following example.

Continuing the above example, assume each intermediate rule in RS_q is to be encoded as a bit string. The rule r'_{q1} is first encoded as shown in Fig. 3.

Since feature *Outlook* in r'_{q1} has three possible test values {*Sunny*, *Cloudy*, *Rain*} which mean "*Outlook* is *Sunny* or *Cloudy* or *Rain*", the test of *Outlook* is then encoded as "111". Similarly, *Humidity* is encoded as "11". *Temperature* has only one test value, *Mild*, and is then encoded as "010". Also, *Wind* is encoded as "01". As a result, each intermediate rule in RS_q is encoded into a chromosomes as shown in Fig. 4.

After knowledge encoding, the genetic process chooses bit-string rules for "mating", gradually creating good offspring rules. The offspring rules then undergo iterative "evolution" until an optimal or a nearly optimal set of rules is found.

4. Knowledge-integration phase

In the knowledge-integration phase, genetic operations and credit assignment are applied at the rule level. All the rule strings among various rule sets together form an initial rule population for later integration. Good rules are then selected for genetic operations to produce better offspring rules. The genetic process runs generation after generation until certain criteria (such as a given number of generations, given processing time, or convergence of fitness values) have been met. After evolution, all the rules in a population are then combined to form a resulting rule set. Details are as follows.

4.1. Initial population

The GA requires a population of individuals to be initialized and updated during the evolution process. In our approach, the initial set of bit strings for rules comes from

	Outlook	***Temperature***	***Humidity***	***Wind***	***Sports***
r'_{q1}	111	010	11	01	100
r'_{q2}	100	100	11	11	010
r'_{q3}	001	100	11	11	001

Fig. 4. Bit-string representation of RS_q.

Table 1
A set of test objects

Case	Outlook			Temperature			Humidity		Wind		Sports		
	Sunny	Cloud	Rain	Hot	Mild	Cool	Humid	Normal	Windy	Not-windy	Volleyball	Swimming	W-lifting
1	1	0	0	1	0	0	1	0	0	1	0	1	0
2	1	0	0	1	0	0	0	1	0	1	1	0	0
3	0	1	0	1	0	0	0	1	0	1	0	1	0
4	0	1	0	0	1	0	0	1	0	1	1	0	0
5	0	0	1	1	0	0	0	1	1	0	0	0	1
6	0	1	0	0	0	1	1	0	0	1	0	0	1
7	0	0	1	0	0	1	0	1	0	1	0	0	1
8	0	1	0	0	0	1	0	1	0	1	1	0	0
9	1	0	0	1	0	0	1	0	1	0	0	1	0
10	1	0	0	0	0	1	0	1	1	0	0	0	1
11	1	0	0	1	0	0	1	0	0	1	0	1	0
12	0	1	0	0	1	0	0	1	0	1	1	0	0
13	1	0	0	0	1	0	0	1	1	0	0	0	1
14	0	1	0	0	1	0	0	1	1	0	0	0	1
15	0	0	1	0	0	1	1	0	1	0	0	0	1
16	1	0	0	0	1	0	0	1	0	1	1	0	0

multiple knowledge sources. Each individual within the population is a rule, and is of fixed length. If all of rule sets have k rules, then the initial population size is k.

4.2. Fitness and selection

In order to develop a "good" knowledge base from an initial rule population, the GA selects *parent* rules with high fitness values for mating. An objective evaluation function and a set of test objects, including documentary evidence, instances or historical records, are used to qualify the rules. Three important factors including accuracy, utility, and coverage of the resulting rules are considered in the evaluation (Yuan & Zhuang, 1996). Let U be the set of test objects. The accuracy of a rule r_i is evaluated using test objects as follows:

$$Accuracy(r_i) = \frac{|\Omega_{r_i}^U|}{|\Omega_{r_i}^U| + |\not{\Omega}_{r_i}^U|},$$

where $\Omega_{r_i}^U$ is the set of test objects in U correctly predicted by r_i, $\not{\Omega}_{r_i}^U$ is the set of test objects in U wrongly predicted by r_i, and $|\Omega_{r_i}^U|$ and $|\not{\Omega}_{r_i}^U|$ denote their cardinality. A test object is wrongly predicted by a rule r_i if it is matched by r_i and its original class is not equal to the class determined by r_i. Obviously, the higher the accuracy of a rule is, the better this rule is.

Since an object may be simultaneously classified by several rules, the utility for each rule is different and must be measured. If an object is correctly predicted by only one rule, this rule is then necessary to correctly classify the object, and its utility value equals 1. If an object is correctly classified by m rules, these rules then share the utility and each rule's utility value is only $1/m$. The total utility of a rule is then evaluated by the sum of its individual utility values for the set of test objects. The utility of a rule (r_i) is defined as follows:

$$Utility(r_i) = \sum_{e \in U} \frac{\Phi(r_i, e)}{\sum_{r} \Phi(r_i, e)},$$

where U is the set of test objects, e is a test object in U, r is a rule in the current population, and

$$\Phi(r, e) = \begin{cases} 1 & \text{if } e \text{ is correctly classified by a rule } r; \\ 0 & \text{otherwise.} \end{cases}$$

Finally, the coverage of a derived rule (r_i) is defined as follows:

$$Coverage(r_i) = |\Omega_{r_i}^U| + |\not{\Omega}_{r_i}^U|.$$

The coverage of a rule r_i is then the number of objects in U matched by the rule condition. Obviously, the larger the coverage, the more general the rule. The three quality functions mentioned above are then considered to determine the fitness values of the rules. Although high accuracy, utility, and coverage are desired, these cannot usually be satisfied at the same time. Trade-offs between them must be made.

Our purpose is to obtain a concise set of rules with high accuracy. Thus, we first sort the rules in descending order of their products of accuracy and utility. The rule with the largest product is first considered to calculate its coverage, and all the test objects covered by this rule are removed from the object set. The fitness of this rule is evaluated according to its accuracy, utility, and coverage values. The same procedure is repeated to calculate the coverage and the fitness value of the next rule based on the remaining object set until the rules in the population are all evaluated. The simultaneous cooperation and competition of individual

Table 2
Redundancy, subsumption and contradiction in the above example

Rule No.	Rule No.	Relationship
1	9	Redundancy
2	7	Redundancy
3	8	Redundancy
4	10	Subsumption
5	11	Redundancy
6	12	Subsumption
5	13	Contradiction
11	13	Contradiction

rules within the population can thus be considered. The evaluation procedure is described as follows:

Step 1. Let the set of test objects be *U*.
Step 2. For each rule r_i in the population, calculate *Accuracy*(r_i) and *Utility*(r_i).
Step 3. Sort the rules in descending order of products of accuracy and utility values.
Step 4. Calculate the coverage value (for the object set *U*) of the first rule in the sorted list, and set the fitness value of the rule as the product of the accuracy, the utility and the coverage.
Step 5. Remove the objects covered by the first rule, say r_j, in the sorted rule list from the current object set *U*, i.e. $U = U - \Omega_{r_j}^{U} - \not\Omega_{r_j}^{U}$.
Step 6. Remove the first rule from the sorted list.
Step 7. Go to Step 4 until the sorted list is empty.

Below, an example is given to illustrate this fitness evaluation process. Continuing the above example, assume a set of test objects is shown in Table 1 (Yuan & Shaw, 1995).

Also assume the following thirteen initial rules have been obtained from different knowledge sources:

r_1: IF (*Temperature* is *Mild*) and (*Wind* is *Not-windy*) then *Volleyball*;
r_2: IF (*Temperature* is *Hot*) and (*Outlook* is *Cloudy*) then *Swimming*;
r_3: IF (*Temperature* is *Hot*) and (*Outlook* is *Sunny*) then *Swimming*;
r_4: IF (*Temperature* is *Hot*) and (*Outlook* is *Rain*) then *Weight-lifting*;
r_5: IF (*Temperature* is *Cool*) then *Weight-lifting*;
r_6: IF (*Temperature* is *Mild*) and (*Wind* is *Windy*) then *Weight-lifting*;
r_7: IF (*Temperature* is *Hot*) and (*Outlook* is *Cloudy*) then *Swimming*;
r_8: IF (*Temperature* is *Hot*) and (*Outlook* is *Sunny*) then *Swimming*;
r_9: IF (*Temperature* is *Mild*) and (*Wind* is *Not-windy*) then *Volleyball*;
r_{10}: IF (*Outlook* is *Rain*) then *Weight-lifting*;
r_{11}: IF (*Temperature* is *Cool*) then *Weight-lifting*;
r_{12}: IF (*Wind* is *Windy*) then Weight-lifting;
r_{13}: IF (*Temperature* is *Cool*) then *Volleyball*;

Redundancy, subsumption and contradiction exist among these rules and are listed in Table 2.

The accuracy and the utility of each rule in the population are first calculated. The results are shown in the columns "accuracy" and "utility" of Table 3.

The rules are thus sorted in descending order of products of their accuracy and utility values. The list of sorted rules is thus {12,1,5,11,9,3,8,6,10,2,7,4,13}. Since rule 12 has the largest product of accuracy and utility (i.e. $0.83 * 1.916 = 1.59$), its coverage and its fitness are first calculated as $|\{5,9,10,13,14,15\}| (= 6)$ and $0.83 * 1.916 * 6 (= 7.915)$. The objects 5,9,10,13,14, and 15 covered by rule 12 are then removed from the object set, and the remaining object set becomes {1,2,3,4,6,7,8,11,12,16}. Also, rule 12 is removed from the sorted list after its fitness evaluation is finished.

The same procedure is repeated to select the rule with the largest product of accuracy and utility values from the new rule list {1,9,5,11,3,8,6,10,2,7,4,13}, and to calculate its coverage and the fitness value based on the remaining object

Table 3
The evaluation process for the above example

Rules	Accuracy (%)	Utility	Accuracy * utility	Coverage	Fitness	Ω_{ri}	$\not\Omega_{ri}$	$U = U - \Omega_{r_i} - \not\Omega_{r_i}$
12	83	1.916	1.59	6	7.915	{5,10,13,14,15}	{9}	{1,2,3,4,6,7,8,11,12,16}
1	100	1.5	1.5	3	4.5	{4,12,16}	–	{1,2,3,6,7,8,11}
9	100	1.5	1.5	0	0	–	–	{1,2,3,6,7,8,11}
5	80	1.416	1.13	3	3.39	{6,7}	{8}	{1,2,3,11}
11	80	1.416	1.13	0	0	–	–	{1,2,3,11}
3	75	1.5	1.125	3	3.375	{1,11}	{2}	{3}
8	75	1.5	1.125	0	0	–	–	{3}
6	100	1	1.0	0	0	–	–	{3}
10	100	0.916	0.916	0	0	–	–	{3}
2	100	0.5	0.5	1	0.5	{3}	–	–
7	100	0.5	0.5	0	0	–	–	–
4	100	0.333	0.333	0	0	–	–	–
13	20	1.0	0.2	0	0	–	–	–

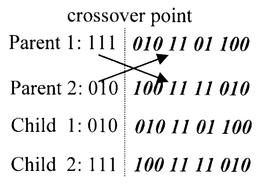
crossover point
Parent 1: 111 010 11 01 100
Parent 2: 010 100 11 11 010
Child 1: 010 010 11 01 100
Child 2: 111 100 11 11 010

Fig. 5. Crossover operation for the above example.

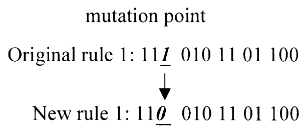
mutation point
Original rule 1: 111 010 11 01 100
New rule 1: 110 010 11 01 100

Fig. 6. Mutation operation for the above example.

Table 4
Numbers of feature values and class patterns in the brain tumor diagnosis

Feature	Number of values	Feature	Number of values
Sex	2	Enhancement degree	4
Location	44	Enhancement appearance	9
Precontrast	6	General appearance	9
Calcification	4	Bone change	6
Edema	4	Mass effect	3
Shape edema	5	Hydrocephalus	3
Number of classes: 6			

set {1,2,3,4,6,7,8,11,12,16}. The process will continue until the rules in the population are all evaluated. The evaluation process is shown in Table 3.

After the evaluation process is performed, rules with low fitness values are eliminated. For example, rules 4,6,7,8,9,10,11, and 13 with fitness values 0 are eliminated. After these rules are deleted, all the redundancy, subsumption, and contradiction listed in Table 2 disappear. The fitness evaluation process can thus generate a concise set of accurate rules from the population.

4.3. Genetic operators

Crossover and *mutation* operators are applied to rules for knowledge integration. The crossover operation is the same as that in the simple genetic algorithm (SGA) proposed by Holland (Holland, 1975). An example of a crossover operation is given below.

Continuing the above example, assume r_1 and r_7 are chosen as the parents for crossover. Assume the crossover point is set at the third gene. The crossover operation is shown as in Fig. 5.

The two newly generated offspring rules are then:

Child *1*: IF (*Outlook* is *Cloudy*) and (*Temperature* is *Mild*) and (*Wind* is *Not-windy*) then *Volleyball*.
Child *2*: IF (*Temperature* is *Hot*) then *Swimming*.

The mutation operator is the same as that in the SGA proposed by Holland. It randomly changes some elements in a selected rule and leads to additional genetic diversity to help the process escape from local-optimum "traps". An example of a mutation operation is given below.

Continuing the above example, assume r_1 is chosen for mutation. Assume the mutation point is set at the third gene. The mutation operation is shown as in Fig. 6.

5. Experimental results

The brain tumor diagnostic problem (Wang, Tseng & Hong, 1995) was used to test the performance of the proposed approach. The 504 cases used in these experiments were obtained from Veterans General Hospital (VGH) in Taipei, Taiwan (Wang et al., 1995). The goal was to identify one of six possible classes of brain tumors, including *Pituitary Adenoma*, *Meningioma*, *Medulloblastoma*, *Glioblastoma*, *Astrocytoma*, and *Anaplastic Protoplasmic Astrocytoma*, which are frequently found in Taiwan. The numbers of possible feature values and class patterns are shown in Table 4.

The 504 cases were first divided into two groups, a training set and a test set. The training set was used to evaluate the fitness values of rule sets during the integration process; the test set acted as input events to test the resulting rule set, and the percentage of correct predictions was recorded. In each run, 70% of the brain tumor cases were selected at random for training, and the remaining 30% of the cases were used for testing. Ten initial rule sets were obtained from different groups of experts at VGH (Wang et al., 1995; Wang et al., 1996), or derived via machine-learning approaches such as PRISM (Cendrowska, 1987), CN2 (Clark & Niblett, 1989) and ID3 (Quinlan, 1986). Each rule, consisting of twelve feature tests and a class pattern, was encoded into a bit string of 105 bits long. The accuracy

Table 5
The accuracy of the ten initial rule sets

Rule sets	Accuracy (%)	No. of rules	Rule sets	Accuracy (%)	No. of rules
Rule set 1	60.03	52	Rule set 6	77.89	56
Rule set 2	79.81	56	Rule set 7	68.53	52
Rule set 3	73.24	56	Rule set 8	72.83	53
Rule set 4	64.74	53	Rule set 9	76.24	56
Rule set 5	58.67	52	Rule set 10	70.19	53

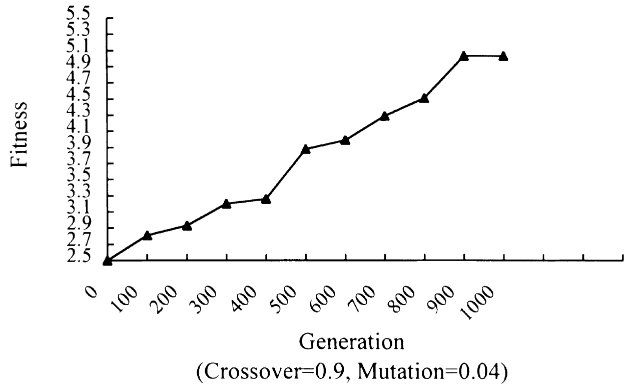

Fig. 7. Relationship between fitness values of the best rule set and generations.

of the ten initial rule sets was measured using the test instances. The results are shown in Table 5.

Although the ten initial rule sets were not accurate enough, however, they could still act as a set of locally optimal solutions that indicate useful information in the search space. Beginning with these rule sets, the proposed genetic knowledge-integration approach could then reach the nearly optimal global solution more rapidly than it could have with nothing to refer to.

In the experiments, *crossover* and *mutation* ratios were set to 0.9 and 0.04, respectively. Fig. 7 shows the relationship between fitness values of the best rule and number of generations.

It is easily seen from Fig. 7 that as the number of generations increased, the resulting fitness value also increased, finally converging to 5.03 approximately. Fig. 8 shows the relationship between average accuracy of the resulting rule sets and number of generations in the knowledge integration.

The knowledge-integration process obtained an accuracy rate of 92.8% after 1000 generations (5533.2 s). The rule number and the complexity of the resulting rule set, averaged over 50 runs, were, respectively, 37 and 0.686.

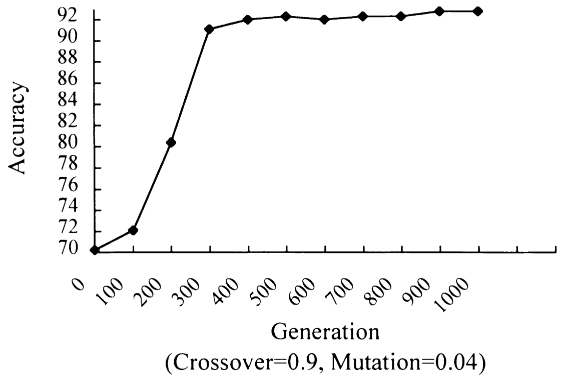

Fig. 8. Relationship between average and number of generations.

6. Conclusions and future work

In this paper, we have shown how the knowledge-integration task can be effectively represented and addressed by the proposed coverage-based genetic strategy. Our approach differs from some other previous knowledge-integration approaches mainly in that it needs no human experts' intervention in the integration process. The time spent by our approach thus depends on computer execution speed, but not on human experts. Much time can thus be saved since experts may be geographically dispersed and experts' discussions are usually time-consuming (Hamilton, 1996; LaSalle & Medsker, 1991). Our approach is also objective, since human experts are not involved in the integration process. Besides, our approach is a scalable integration method that can be applied as well when the number of rule sets to be integrated increases. The CPU time increased is small when compared to human-intervening knowledge-integration approaches.

Although the work presented here shows good results, it is only a beginning. Some future investigations are still needed. For example, the vocabulary problem in the field of knowledge integration remains a troublesome impediment. Our approach assumes that all rule sets use the same vocabulary set. Integrating rule sets having different vocabularies is still full of challenge. Also, several issues in the field of knowledge verification remain unresolved. Our approach addresses three commonly seen issues, redundancy, subsumption, and contradiction. Dealing with other knowledge verification issues in the integration is an interesting topic.

References

Baral, C., Kraus, S., & Minker, J. (1991). Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering, 3* (2), 208–220.

Boose, J. H. (1985). Knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man–Machine Studies, 23,* 495–525.

Boose, J. H., & Bardshaw, J.M. (1987). Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems. *International Journal of Man–Machine Studies, 26,* 3–28.

Carse, B., Fogarty, T. C., & Munro, A. (1996). Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Sets and Systems, 80,* 273–293.

Cendrowska, J. (1987). PRISM: an algorithm for inducing modular rules. *International Journal of Man–Machine Studies, 27,* 349–370.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine learning, 3,* 261–283.

Cordon, O., Herrera, F., & Lozano, M. (1997). On the combination of fuzzy logic and evolutionary computation: a short review and bibliography. In W. Pedrycz, *Fuzzy evolutionary computation* (pp. 33–56). Dordrecht: Kluwer Academic Publishers.

DeJong, K. A., Spears, W. M., & Gordon, D. F. (1993). Using genetic algorithms for concept learning. *Machine learning, 13,* 161–188.

Gaines, B. R. (1989). Integration issues in knowledge supports systems. *International Journal of Man–Machine Studies, 26,* 495–515.

Gaines, B. R., & Shaw, M. L. (1993). Eliciting knowledge and transferring it effectively to a knowledge-based system. *IEEE Transaction on Knowledge and Data Engineering*, *5* (1), 4–14.

Giarratano, J., & Riley, G. (1993). *Expert system principles and programming* (pp. 02116–4324). Boston, MA: PWS Publishing Company.

Hamilton, D. M. (1996). Knowledge acquisition for multiple site, related domain expert systems: Delphi process and application. *Expert Systems With Applications*, *11* (3), 377–389.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*, . Ann Abor, Michigan: University of Michigan Press.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (pp. 1–7). Hillsdale, NJ: Lawrence Erlbaum.

Holland, J. H., & Reitman, J. S. (1983). Cognitive systems based on adaptive algorithms. *Machine learning: an artificial intelligence approach*Los Altos, CA: Morgan Kaufmann.

LaSalle, A. J., & Medsker, L. R. (1991). Computerized conferring for knowledge acquisition from multiple experts. *Expert Systems With Applications*, *3*, 517–522.

Medsker, L., Tan, M., & Turban, E. (1995). Knowledge acquisition from multiple experts: problems and issues. *Expert Systems with Applications*, *9* (1), 35–40.

Quinlan, J. (1986). Induction of decision tree. *Machine learning*, *1*, 81–106.

Smith, S.F. (1980). *A learning system based on genetic adaptive algorithms*. PhD thesis, University of Pittsburgh.

Wang, C. H., Tseng, S. S., & Hong, T. P. (1995). Design of a self-adaptive brain tumor diagnostic system. *Journal of Information Science and Engineering*, *11*, 275–294.

Wang, C. H., Hong, T. P., & Tseng, S. S. (1996). Self-integrating knowledge-based brain tumor diagnostic system. *Expert Systems with Applications*, *11* (3), 351–360.

Wang, C. H., Hong, T. P., & Tseng, S. S. (1997). Knowledge integration by genetic algorithms. *Proceedings of the Seventh International Fuzzy Systems Association World Congress*, *2*, 404–408.

Wang, C. H., Hong, T. P., & Tseng, S. S. (1998). A genetic fuzzy-knowledge integration framework. *The Seventh International Conference of Fuzzy Systems*, 1194–1199.

Wang, C. H., Hong, T. P., Tseng, S. S., & Liao, C. M. (1998). Automatically integrating multiple rule sets in a distributed knowledge environment. *IEEE Transactions on Systems, Man, and Cybernetics-Part C, 28* (3), 471–476.

Wang, C. H., Hong, T. P., & Tseng, S. S. (2000). *Integrating membership functions and fuzzy rule sets from multiple knowledge sources, Fuzzy Sets and Systems, 112*, 141–154.

Wilson, S. W. (1994). ZCS: zeroth level classifier system. *Evolutionary Computation*, *2* (1), 1–18.

Yuan, Y., & Shaw, M. J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, *69*, 125–139.

Yuan, Y., & Zhuang, H. (1996). A genetic algorithm for generating fuzzy classification rules. *Fuzzy Sets and Systems*, *84*, 1–19.