of manipulators is small. However, with an increase in the number of degrees of freedom of the manipulators, the parallel algorithms will be more efficient than the serial algorithms. For example, the serial algorithm of [5] is more efficient than the parallel algorithm of [14], however, when $n = 6$, the operations of computation of [14] is 556 while [5] is 869.

2) Compared with other algorithms, the parallel algorithm using three processors is more efficient.
3) Because the formulation developed in this paper possesses good parallelism, the further improvement in computational efficiency could easily be achieved by increasing the processors, while it is not the same case based on the recursive Newton–Euler formulation [10].

REFERENCES

[1] C. A. Balafoutis and R. V. Patel, *Dynamics Analysis of Robot Manipulators—A Cartesian Tensor Approach.* Norwell, MA: Kluwer, 1991.

[2] J. Y. S. Luh, M. W. Walker, and R. P. Paul, "On-line computational scheme for mechanical manipulator," *Trans. ASME J. Dynam. Syst. Meas. Contr.*, vol. 120, pp. 69–76, 1980.

[3] W. S. Wang, K. K. Chen, Y. S. Lai, and C. H. Liu, "Implementation of a multiprocessor system for real-time inverse dynamics computation," in *IEEE Conf. Robot. Automat.*, vol. 5, pp. 1174–1179, Dec. 1989.

[4] J. M. Hollerbach, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, pp. 730–736, Nov. 1980.

[5] X. G. He and A. A. Goldenberg, "An algorithm for efficient computation of dynamics of robotic manipulators," *J. Robot. Syst.*, vol. 7, no. 5, pp. 689–702, 1990.

[6] C. S. G. Lee and P. R. Chang, "Efficient parallel algorithm for robot inverse dynamics computation," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, pp. 532–542, Apr. 1986.

[7] A. Fijiany and A. K. Bejczy, "A class of parallel algorithms for computation of the manipulator inertia matrix," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 600–615, Oct. 1989.

[8] J. Y. S. Luh and C. S. Lin, "Scheduling of parallel computation for a computer-controlled mechanical manipulator," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-12, pp. 214–234, Feb. 1982.

[9] L. H. Lathrop, "Parallelism in manipulator dynamics," *Int. J. Robot. Res.*, vol. 4, no. 2, pp. 80–102, 1985.

[10] H. Kasahara and S. Narita, "Parallel processing of robot-arm control computation on a multiprocessor system," *IEEE Trans. Robot. Automat.*, vol. RA-1, pp. 104–113, Apr. 1985.

[11] M. Vukobratovic, N. Kircaski, and S. G. Li, "An approach to parallel processing of dynamic robot models," *Int. J. Robot. Res.*, vol. 7, no. 2, pp. 64–71, 1988.

[12] C. L. Chen, C. S. G. Lee, and E. S. H. Hou, "Efficient scheduling algorithms of robot inverse dynamics computation on a multiprocessor system," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 729–743, May 1988.

[13] C. S. G. Lee and C. L. Chen, "Efficient mapping algorithms for scheduling robot inverse dynamics computation on a multiprocessor system," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 582–595, Mar. 1990.

[14] K. Hashimoto and H. Kimura, "A new parallel algorithm for inverse dynamics," *Int. J. Robot. Res.*, vol. 8, no. 1, pp. 63–76, 1989.

[15] K. Hashimoto, K. Ohashi, and H. Kimura, "A implementation of a parallel algorithm for real-time model-based control on a network of microprocessors," *Int. J. Robot. Res.*, vol. 9, no. 6, pp. 37–47, 1990.

[16] E. Binder and J. H. Herzog, "Distributed computer architecture and fast parallel algorithms in real-time robot control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, pp. 543–549, Apr. 1986.

[17] Y. F. Zheng and H. Hemami, "Computation of multibody system dynamics by a multiprocessor scheme," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, pp. 102–110, Jan. 1986.

[18] J. J. Craig, *Introduction to Robotics: Mechanics and Control.* New York: Addison-Wesley, 1986.

Automatically Integrating Multiple Rule Sets in a Distributed-Knowledge Environment

Ching-Hung Wang, Tzung-Pei Hong, Shian-Shyong Tseng, and Chih-Mao Liao

***Abstract*— In this paper, an actual knowledge application is made by means of evolution paradigms in terms of knowledge acquisition. An automatic knowledge integration approach in a distributed-knowledge environment is thus proposed to integrate multiple rule sets into a single effective rule set. The proposed approach consists of two phases: knowledge encoding and knowledge integration. In the encoding phase, each knowledge input is translated and expressed as a rule set, then encoded as a bit string. The combined bit strings from multiple knowledge inputs form an initial knowledge population, which is then ready for integration. In the knowledge integration phase, a genetic search technique generates an optimal or nearly optimal rule set from these initial knowledge-input strings. Finally, experimental results from diagnosis of brain tumors show that the rule set derived by the proposed approach is much more accurate than each initial rule set.**

I. INTRODUCTION

Developing an expert system requires construction of a complete, consistent, and unambiguous knowledge base [2], [16]. The knowledge required to develop a knowledge-based system is often distributed among groups of experts rather than being available for elicitation from a single expert [11]. Acquiring and integrating multiple knowledge inputs from many experts or by various knowledge-acquisition techniques [6], [7], [17], [21] thus plays an important role in building effective knowledge-based systems. Recently, knowledge acquisition and integration systems [3], [12], [19], based on the *Personal Constructs Psychology (PCP)* model [15] or *Integrity Constraints* [2], [16], have been successfully applied. Examples include AQUINAS [4], ETS [5], KSSO [12], and KITTEN [19]. These knowledge-integration methods then present the following problems.

1) Domain experts must intervene during integration to resolve conflicts and contradictions.
2) Integration is time-consuming (requiring weeks or months).
3) The more knowledge sources consulted, the more difficult and complex the integration.

In order to overcome these problems, we propose a genetic knowledge-integration method that automatically combines multiple rule sets into one integrated knowledge base by means of evolutionary knowledge acquisition paradigms. Each rule set is encoded as a bit string and evaluated by an evaluation function. The proposed method then chooses good rule sets according to their fitness values and mates them. Domain experts need not intervene in the integration process. Finally, experimental results show that the proposed approach can greatly improve the knowledge base. Our knowledge integration is thus a successful application of genetic algorithms. Specifically, our

Manuscript received November 7, 1995; revised July 1, 1996, February 8, 1997, August 5, 1997, and November 2, 1997. This work was supported by the National Science Council of the Republic of China.

C.-H. Wang is with the Chunghwa Telecommunication Laboratories, Chung-Li, 32617 Taiwan, R.O.C. (e-mail: ching@ms.tl.gov.tw).

T. P. Hong is with the Department of Information Management, I-Shou University, Kaohsiung, 84008 Taiwan, ROC (e-mail: tphong@csa500.isu.edu.tw).

S.-S. Tseng and C.-M. Liao are with the Institute of Computer and Information Science, National Chiao-Tung University, Hsin-Chu 30050, Taiwan, R.O.C. (e-mail: sstseng@cis.nctu.edu.tw).

Publisher Item Identifier S 1094-6977(98)03899-1.

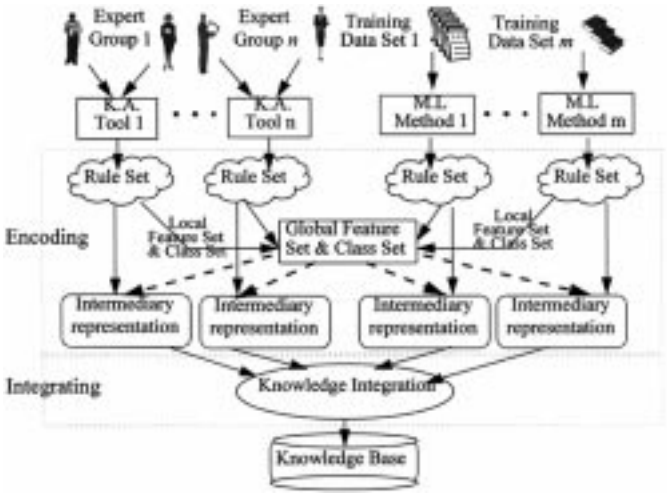

Fig. 1. Knowledge-integration scheme in a distributed-knowledge environment.

approach can effectively integrate multiple knowledge sources in an environment with good communication facilities of networks.

The remainder of this paper is organized as follows. An automatic knowledge-integration approach for a distributed-knowledge environment is proposed in Section II. Experiments on brain tumor diagnosis are reported in Section III. Conclusions are given in Section IV.

II. Knowledge Integration in a Distributed-Knowledge Environment

In this section, we propose a genetic knowledge-integration approach that rapidly constructs a knowledge-based system without human intervention by integrating knowledge from multiple sources. Here, we assume 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. Fig. 1 shows the proposed knowledge-integration scheme. A genetic algorithm is used to maintain a population of possible rule sets and search for the best integrated one.

Our knowledge integration consists of two processes: ***encoding*** and ***integration***. The encoding process transforms each rule set into a bit-string structure. The integration process chooses bit-string rule sets for "mating," gradually creating good offspring rule sets. The offspring rule sets then undergo recursive "evolution" until termination criteria are met. The best bit string rule set in the population is then output and translated into if–then rules. The encoding process is first explained below.

A. Knowledge Encoding

Since rule sets derived from different knowledge sources generally differ in syntax and size, designing an appropriate data structure to accommodate these rule sets is crucial. Several strategies have been proposed to represent rule-set knowledge structures for conceptual learning [3], [9], [10], [14]. For example, the **Michigan approach** [3] encodes individual rules into fixed-length bit strings, with each individual in the population representing a rule. Another, the **Pittsburgh approach** [9], encodes rule sets into variable-length bit strings, with each individual in the population representing a rule set. Since multiple rule sets must be combined, and the rule sets are derived from different sources, representation of variable-length rule sets is preferred in this paper to preserve the syntactic and semantic constraints of variable-length rule sets. We thus encode knowledge using the Pittsburgh approach. The rule sets from different sources must, however, be translated into a uniform syntactical representation before being encoded. The steps for translation of rule sets are described below.

1) Collect the features and possible values from the conditional parts of the rule sets. An elementary feature is a unit condition that cannot be decomposed into simpler ones. All features comprise a feature set.
2) Collect classes from the conclusion parts of the rule sets. An elementary class is a unit conclusion that cannot be decomposed into simpler ones. All classes comprise a class set.
3) Translate each rule into an intermediary representation that retains its essential syntax and semantics. If some features in the feature set are not used by the rule, *dummy* tests are inserted into the condition part of the rule. Each rule in the intermediary representation is then composed of ***N feature tests*** and one ***class pattern***, where N is the number of global features collected.
4) If the *feature tests* or *class patterns* are numerical, they are first discretized into a number of possible regions [18]. Each feature test is then encoded into a fixed-length binary string, with length equal to the number of possible test values. Each bit thus represents a possible value. 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.
5) For each rule set, concatenate all its rule substrings. Since different rule sets might have different numbers of rules, the lengths of the rule sets might be different.

An example that demonstrates the encoding process is shown below.

Example 1: Assume in diagnosing brain tumors, two classes {Adenoma, Meningioma} are to be distinguished using three features {Location, Calcification, Edema}. Assume Feature *Location* has three possible values {brain surface, sellar, brain stem}, Feature *Calcification* has four possible values {none, marginal, vascular-like, lumpy}, and Feature *Edema* has three possible values {none, <2 cm, <0.5 hemisphere}. Also assume that a rule set RS_i from a knowledge source has only these two rules:

R_1 if (Location = sellar) and (Calcification = no), then Class is Adenoma;
R_2 if (Location = brain surface) and (Edema < 2 cm), then Class is Meningioma.

The intermediary representation of R_1 and R_2 would then be constructed as follows:

R_1' if (Location = sellar) and (Calcification = no) and (Edema = no or Edema < 2 cm or Edema < 0.5 hemisphere), then Class is Adenoma;
R_2' if (Location = brain surface) and (Calcification = no or Calcification = marginal or Calcification = vascular like or Calcification = lumpy) and (Edema < 2 cm), then Class is Meningioma.

The tests with underlines are dummy tests. Also, R_1 and R_2 are logically equivalent to R_1' and R_2'. Using the intermediary form, we encode each feature test into a fixed-length binary string. For example, the set of legal values for feature location is {brain surface, sellar, brain stem}. Three bits are then used to represent this feature. The bit string 101 would represent the test for Location being "brain surface" or "brain stem." As a result, the above rules are, respectively, encoded as follows:

Rule	*Location*	*Calcification*	*Edema*	*Class*
R_1'	010	1000	111	10
R_2'	100	1111	010	01

Finally, rule set RS_i is encoded as "01010001111010011 1101001."

As Example 1 shows, we can encode and translate rule sets into bit strings with a uniform syntax for integration.

B. Knowledge Integration

In our approach, the initial set of bit strings for rule sets comes from multiple knowledge sources. Each rule set represents one individual in the initial population. In order to develop a "good" knowledge base from an initial population of rule sets, the genetic algorithm selects *parent* rule sets with high fitness values for mating. An objective evaluation function and a set of training instances then qualify the rule set. The training instances are examples actually occurring in the applications and can also be gathered from the distributed environment. Two important factors are used in evaluating derived rule sets, the accuracy and the complexity of the resulting knowledge structure. Accuracy is evaluated using training instances as in (1), shown at the bottom of the page. The complexity of the resulting rule set (RS) is the ratio of rule increase, defined as in (2), shown at the bottom of the page, where RS_i is the initial rule set, and m is the number of initial rule sets. Accuracy and complexity are combined to represent the fitness value of the rule set. The evaluation function is defined as follows:

$$\text{fitness}(RS) = \frac{[\text{Accuracy}(RS)]}{[\text{Complexity}(RS)]^{\alpha}} \tag{3}$$

where α is a control parameter, representing a tradeoff between accuracy and complexity. The fitness function can also reduce the impact of noisy information that causes rule-set overfitting (excessive complexity) [7].

C. Genetic Operators

Genetic operators are very important to the success of specific GA applications. There are generally two ways of employing genetic operators [10]: conventional mutation and crossover operators only may be used or new genetic operators may be added to these. Each has its advantages and disadvantages. By designing new genetic operators, the genetic process can take domain-specific characteristics into consideration, thus making the results closer to those desired. This however takes more execution time than using only the original operators. In this paper, both ways are examined and compared. Two fundamental genetic operators (*crossover*, *mutation*) and two domain-specific operators (*fusion*, *fission*) are used in the genetic algorithm.

The crossover operator used in this paper selects crossover points differently from that in the simple genetic algorithm. The crossover operator in the simple genetic algorithm chooses the same points for both parent chromosomes, but, the crossover operator here need not use the same point positions for both parent chromosomes. The crossover points may occur within rule strings or at rule boundaries. The only requirement for crossover points is that they "match up semantically."

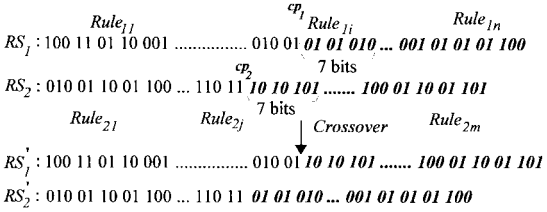

Fig. 2. Example of crossover.

The crossover operator takes two parent rule sets and swaps parts of their genes to produce offspring rule sets. If the genes swapped are desirable, the offspring rule sets will inherit these advantages from their parents and will survive after this generation. Thus, the resulting rule sets will be closer to the one really desired from generation to generation. The detailed procedure is shown below.

1) Select a crossover point in one of the parents at random.
2) If the chosen point occurs at some rule boundary, the crossover point in the other parent must also be at a rule boundary. Otherwise, the point may be within the rule string p bits to left of a rule boundary. The crossover point for the other parent must also be within the rule string and p bits to left of some rule boundary.
3) Cross the genes of the parents according to the crossover points.
4) Generate new offsprings.

Example 2: Assume that parent rule sets RS_1 and RS_2 (shown in Fig. 2), respectively, contain n and m rules for classifying training instances with four features (F_1, F_2, F_3, and F_4). Feature F_1 has three possible values, features F_2, F_3, and F_4 all have two possible values. Three classes are to be determined. A crossover operation with crossover points at cp_1 and cp_2 is shown in Fig. 2.

The mutation operator is the same as the that used in the simple genetic algorithm. It randomly changes some elements in a selected rule set, leading additional genetic diversity to help the integration process escape from local-optimum "traps."

Problems such as *redundancy* [13], *subsumption* [13], *contradiction* [13], and *misclassification* [13] do arise when integrating rule sets. We thus propose two domain-specific operators, *fusion* and *fission*, to solve these problems. Fusion emulates natural evolution's chromosomal genetic folding phenomenon to eliminate redundancy and subsumption. Assume that redundant rules $Rule_{ki}$ and $Rule_{kj}$ belong to rule set RS_k. The redundancy relationship between $Rule_{ki}$ and $Rule_{kj}$ is found by matching strings. Below, an example is given to illustrate how fusion deals with redundancy.

Example 3: Assume that redundant rules $Rule_{ki}$ and $Rule_{kj}$ exist in RS_k. To remove the redundancy, fusion folds the redundant rules, as shown in Fig. 3.

$$\text{Accuracy}(RS) = \frac{\text{the total number of training instances correctly matched by } RS}{\text{the total number of training instances}} \tag{1}$$

$$\text{Complexity}(RS) = \frac{\text{Number of rules within the integrated rule set } RS}{\left[\sum_{i=1}^{m} (\text{Number of rules within initial } RS_i)\right] \Big/ m} \tag{2}$$

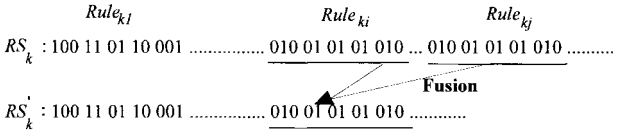

Fig. 3. Fusion operation for eliminating redundancy.

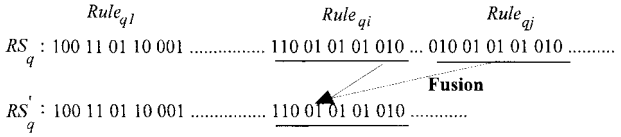

Fig. 4. Fusion operation for eliminating subsumption.

Since redundancy is a special case of subsumption, the fusion function can be extended to solve subsumption problems. Assume that in the rule set RS_q, $Rule_{qi}$ is subsumed by $Rule_{qj}$. The logical "OR" operation can be used to find subsumptive relationships. If a string resulting from an "OR" operation between $Rule_{qi}$ and $Rule_{qj}$ is the same as $Rule_{qi}$, rule $Rule_{qi}$ subsumes rule $Rule_{qj}$. The fusion operator eliminates this by folding the subsumptive rules together. Below, an example is given to illustrate how fusion deals with subsumption.

Example 4: Assume that in rule set RS_q, $Rule_{qi}$ subsumes $Rule_{qj}$. The fusion operator folds the subsumptive rules together, removing the subsumption as shown in Fig. 4.

The fusion operator thus reduces rule-set complexity by eliminating redundancy and subsumption.

Misclassifications and contradictions may also occur in rule sets. We propose a fission operator that emulates natural evolution's chromosomal *schizogenesis* to eliminate misclassifications and contradictions. A misclassification occurs when rules classify training instances incorrectly. Fission selects the "closest" *near-miss* rule [21] to specialize the wrongly classified training instance. Assume rules $Rule_{ki}, \cdots, Rule_{kj}, \cdots, Rule_{kn}$ all misclassify training instance $\boldsymbol{I}$. The closest near-miss relationship among these rules with respect to $\boldsymbol{I}$ can be found by string-matching. If $Rule_{kj}$ has the fewest differing string bits when matched with training instance $\boldsymbol{I}$, then $Rule_{kj}$ is the closest near-miss to training instance $\boldsymbol{I}$. Fission specializes $Rule_{kj}$ into more specific rules that do not include this instance. Instance $\boldsymbol{I}$ is also transformed into a rule and inserted into the rule set. An example of handling misclassification is given below.

Example 5: Assume that rule $Rule_{kj}$ in rule set RS_k misclassifies training instance $\boldsymbol{I}$ and is the closest near-miss to this instance. Assume that instance $\boldsymbol{I}$ has four features F_1, F_2, F_3, $F4$, one *class pattern*, and is encoded as 100, 10, 01, 10, 010. The execution of the fission operator proceeds as shown in Fig. 5.

Two kinds of contradiction may occur in a rule set. The first occurs when two rules with the same feature values point to different classes; the second occurs when a rule points to two or more classes simultaneously. The first case of contradiction is removed by the crossover and mutation operators; that is, the bit strings of conflicting rules may be altered or rearranged to remove the contradiction. In the second case, the fission operator is used to split the rule into more specific ones, each pointing to only one class. These new rules will still contradict each other, but have been reduced to the first class of contradiction, and can then be removed by the crossover and mutation operators. An example of resolving a contradiction is given below.

Example 6: Assume that the rule set RS_k contains the rule $Rule_{ki}$, which points to two conclusions. To resolve the contradiction in $Rule_{ki}$, fission specializes rule $Rule_{ki}$ into rules $Rule'_{ki}$ and $Rule''_{ki}$, each having only one conclusion, as shown in Fig. 6.

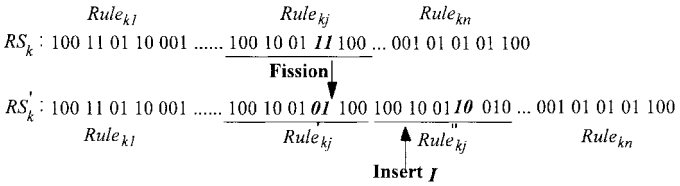

Fig. 5. Fission operation for eliminating misclassification.

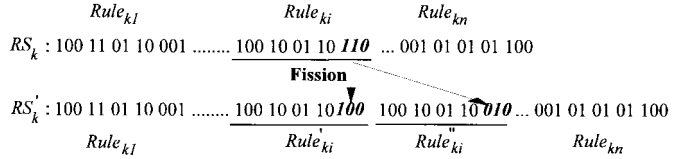

Fig. 6. Fission operation for eliminating contradiction.

TABLE I
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

The fission operator thus acts as a specialization operation that resolves misclassification and contradiction problems and promotes rule accuracy.

Although fusion and fission may raise rule-set accuracy, they cause great computational burdens if performed whenever a new chromosome is generated. Operation probabilities for these two operators are thus set to achieve a tradeoff between time complexity and accuracy. These two operators are then performed on a new chromosome with the given probabilities.

III. EXPERIMENTAL RESULTS

Brain tumor diagnosis [20] was used as the problem domain to test the performance of the proposed knowledge-integration approach. The 504 cases used in these experiments were obtained from Veterans General Hospital (VGH), Taipei, Taiwan [20]. The goal of the experiments was to identify one of six possible classes of brain tumors frequently found in Taiwan. 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 of rule sets during the integration process; the test set acted as input events to test the derived 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. Because of incorrect recording or transcription errors, the cases contained noise that led to cases with identical features being classified differently. Ten initial rule sets were obtained from different groups of experts at VGH or derived via machine-learning [6], [7], [17], [20]. Each rule, consisting of twelve feature tests and a class pattern, was encoded into a bit string 105 bits long. The accuracy of the ten initial rule sets was measured using the test instances. The results are shown in Table I.

Although the ten initial rule sets were not accurate enough, they could still however act as a set of locally-optimal solutions that indicate useful information in the search space. Beginning with these

TABLE II
EFFECTIVENESS OF THE DOMAIN-SPECIFIC OPERATORS

Rule Sets	Accuracy	No. of rules	Time (second)	Complexity
Integration without domain-specific operators (2000 generations)	85.83%	168	11264.3	3.1168
Integration with domain-specific operators (700 generations)	87.83%	75	5102.4	1.3914
Integration with domain-specific operators (2000 generations)	91.42%	92	15264.7	1.7086

rule sets, the genetic algorithm could then more rapidly reach the (nearly) global optimal solution than that it could have with nothing to refer to. Of course, each initial rule set could first have been improved by the same GA scheme before integration. We do not however prefer this alternative since it cannot use information from the other rule sets and thus would take more time to get good results. Similarly, we could also have abandoned these initial rule sets and directly applied the genetic algorithm to acquire knowledge from training instances. But the same disadvantages would still have resulted.

In the experiments, the operation frequency to each bit string chosen for crossover, mutation, fusion, and fission operators was set at 0.9, 0.04, 0.01, and 0.01, respectively. The parameter α in the fitness function is used for a tradeoff between accuracy and complexity. If the α value is small, the fitness function then focuses on the classification accuracy. Contrarily, if the α value is large, the fitness function is then dominated by the complexity. Here, α was set at 0.125. Two experiments were made to evaluate the effectiveness of the proposed methods: one with domain-specific operators (fusion and fission) and the other without them. Table II shows a comparison of these two approaches as to accuracy, number of rules in the resulting knowledge base, integration time, and resulting complexity, averaged over 50 runs.

Experimental results show that using the knowledge-integration algorithm with domain-specific operators yields more accurate results than those obtained without domain-specific operators. Domain-specific operators, however, have additional computational burdens. As Table II shows, integration with domain-specific operators takes 15 264.7 s after 2000 generations, but, without domain-specific operators, needs only 11264.3 s. Note that every accuracy rate in Table II is higher than that of any initial rule set in Table I.

Fig. 7 shows the relationship between the generations and the fitness values of the resulting rule set for the proposed approach at different domain-specific operator rates. The more generations there were, the higher the fitness values of the resulting rule set. The integration process initially appeared unstable. Fusion operations reduced the structural complexity of the resulting rule sets, but fission operations increased it. Also, fusion does not affect the accuracy of the resulting rule sets, but fission may increase it. It is thus hard to see any trend for the domain-specific operators at this stage. In the later stages, the integration process stabilizes. The larger the domain-specific operator probabilities, the higher the fitness values of the resulting rule set. Domain-specific operators thus play an important role in the genetic process.

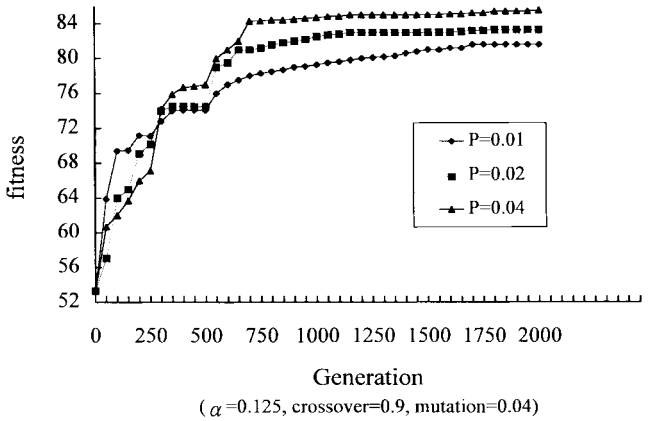

Fig. 7. Relationship between fitness values and generations for different domain-specific operator probabilities.

IV. CONCLUSIONS AND DISCUSSIONS

We have shown how knowledge integration can be effectively represented and addressed by a genetic algorithm with two domain-specific operators. Experimental results showed that our genetic knowledge-integration approach is valuable for combining multiple rule sets in distributed-knowledge environments. Our approach differs from some notable approaches [5], [8], [16], mainly in that it requires no human experts' intervention during integration. Our approach is thus dependent on computer execution speeds, not on human experts. This saves much time since experts may be geographically dispersed, and their deliberations may be very time consuming. Our approach is also scalable and can be used effectively when the number of rule sets to be integrated is large, and integrating large numbers of rule sets may increase the validity of the resulting knowledge base. Our method is also objective since human experts are not involved in the integration process.

Problems such as redundancy, subsumption, misclassification, and contradiction in the integration are solved automatically. Although, the work presented here shows good results, it is only a beginning. Some future investigations are proposed below.

1) Knowledge sources or actual instances may contain fuzzy information in the real world. We are currently studying how to automatically generate the appropriate membership functions and deal with fuzzy knowledge.
2) Many issues in the field of knowledge verification remain unresolved. Our approach addresses four commonly seen issues (redundancy, subsumption, misclassification, and contradiction). Modifying or designing new genetic operators to deal with other knowledge verification issues is another interesting topic.

REFERENCES

[1] T. R. Addis, "Knowledge refining for a diagnostic aid," *Int. J. Man–Mach. Stud.*, vol. 17, pp. 151–164, 1982.
[2] C. Baral, S. Kraus, and J. Minker, "Combining multiple knowledge bases," *IEEE Trans. Knowledge Data Eng.*, vol. 3, pp. 208–220, Mar./Apr. 1991.
[3] L. B. Booker, "Intelligent behavior as an adaptation to the task environment," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1982.
[4] J. H. Boose and J. M. Bardshaw, "Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems," *Int. J. Man–Mach. Stud.*, vol. 26, pp. 3–28, 1987.
[5] J. H. Boose, "A knowledge acquisition program for expert systems based on personal construct psychology," *Int. J. Man–Mach. Stud.*, vol. 23, pp. 495–525, 1985.
[6] J. Cendrowska, "PRISM: An algorithm for inducing modular rules," *Int. J. Man–Mach. Stud.*, vol. 27, pp. 349–370, 1987.
[7] P. Clark and T. Niblett, "The CN2 induction algorithm," *Mach. Learn.*, vol. 3, pp. 261–283, 1989.

[8] D. D. Corkill, K. Q. Gallagher, and K. E. Murray, "GBB: A generic blackboard development system," in *Blackboard Systems*, R. Engelmore and T. Morgan, Eds. New York: Addison-Wesley, 1988, pp. 503–518.
[9] K. A. DeJong, "Learning with genetic algorithm: An overview," *Mach. Learn.*, vol. 3, pp. 121–138, 1988.
[10] K. A. DeJong, W. M. Spears, and D. F. Gordon, "Using genetic algorithms for concept learning," *Mach. Learn.*, vol. 13, pp. 161–188, 1993.
[11] B. R. Gaines, "Integration issues in knowledge supports systems," *Int. J. Man–Mach. Stud.*, vol. 26, pp. 495–515, 1989.
[12] B. R. Gaines and M. L. G. Shaw, "Eliciting knowledge and transferring it effectively to a knowledge-based system," *IEEE Trans. Knowledge Data Eng.*, vol. 5, pp. 4–14, Jan./Feb. 1993.
[13] J. Giarratano and G. Riley, *Expert System Principles and Programming.* Boston, MA: PWS, 1993.
[14] C. Z. Janikow, "A knowledge-intensive genetic algorithm for supervised learning," *Mach. Learn.*, vol. 13, pp. 189–228, 1993.
[15] G. A. Kelly, *The Psychology of Personal Constructs.* New York: Norton, 1955.
[16] O. K. Ngwenyama and N. Bryson, "A formal method for analyzing and integrating the rule-sets of multiple experts," *Inf. Syst.*, vol. 17, no. 1, pp. 1–16, 1992.
[17] J. R. Quinlan, "Induction of decision tree," *Mach. Learn.*, vol. 1, pp. 81–106, 1986.
[18] ____, "Decision tree as probabilistic classier," in *Proc. 4th Int. Mach. Learn. Workshop*, 1987, pp. 31–37.
[19] M. L. G. Shaw and B. R. Gaines, "KITTEN: Knowledge initiation and transfer tools for experts and novices," *Int. J. Man–Mach. Stud.*, vol. 27, pp. 251–280, 1987.
[20] C. H. Wang, T. P. Hong, and S. S. Tseng, "Design of a self-adaptive brain tumor diagnostic system," *J. Inf. Sci. Eng.*, vol. 11, pp. 275–294, 1995.
[21] P. H. Winston, *Artificial Intelligence*,, 3rd ed. New York: Addison-Wesley, 1992.

Matching Strengths of Answers in Fuzzy Relational Databases

Ding-An Chiang, Nancy P. Lin, and Chien-Chou Shis

***Abstract*—In this paper, imprecise information is represented by fuzzy disjunctive information, and an extended fuzzy relational model is used to accommodate such information. In the presence of imprecise information, answers to a query can be categorized into two kinds of answers: sure answers and possible answers. To find more likely answers to a given query, we develop a method to measure the matching strength of each tuple as an answer to the query. The quality of an answer is higher in the case where less extra information is required and the more sure information is provided.**

***Index Terms*—Extra information, fuzzy disjunctive information, matching information, self-information.**

I. Introduction

In real-world applications, the motivation of using fuzzy set theory in database systems lies on the need of handling imprecise information. Some authors have used fuzzy set theory to accommodate different types of imprecise information [1]–[4], [9], [15]. Frequently, the major strategy of dealing with imprecise information in fuzzy relational databases is to use a set of values to express imprecise information [1]–[3], [22]. Accordingly, Raju and Majumdar [19] classify fuzzy relational databases into two types: *type-1* fuzzy relational databases and *type-2* fuzzy relational databases. The characteristic of *type-1* relational databases is that the attribute value is conformed and *homogeneous*; each attribute domain can only be a fuzzy set. On the other hand, a *type-2* fuzzy relational database is a *heterogeneous* fuzzy relational database, and each attribute domain can be a set of fuzzy sets.

Some authors have defined different fuzzy relational models to accommodate different types of imprecise information. The attribute values can be a set of values, and the attribute domains may contain different data types in these models [1], [2], [4], [17]. However, information such as "((John), (Engineering), (30 000)) $\vee$ ((John), (Manager), (high, 0.9))" cannot be easily represented by these models. Therefore, to capture more of the meaning of the data, the fuzzy relational model is generalized based on the theory of *type-2* fuzzy relation and the *first-order logic*. Each tuple $\boldsymbol{t}$ will be fuzzy disjunctive information in the proposed model, where fuzzy disjunctive information is a generalization of disjunctive information, which has been studied by many authors in classical relational databases [6]–[8], [13], [26]. For example, John's age is either 18 or 19 and "((John), (Sales), (30 000)) $\vee$ ((John), (Manager), (high, 0.9))" is fuzzy disjunctive information. Consequently, a fuzzy relational database is viewed as a particular kind of first-order logic interpretations, and the query processing is a truth determination of a first-order formula applied to a fuzzy relational database.

With the introduction of imprecise information into a database, we need to consider measures of uncertainty during the query evaluation process, and much research has been done regarding measures of uncertainty. Probability theory is the classical means of handling uncertainty, and it is useful and successful in many applications. However, the problem of using probability theory is that the *a priori* probability of an object is difficult or impossible to estimate [16]. Therefore, in this paper, we assume that all attributes are independent and identically distributed.

Several researchers have viewed the measure of uncertainty as a measure of fuzziness. De Luca and Termini [14] showed that a measure of uncertainty can be viewed as a measure of fuzziness. Following De Luca and Termini's suggestion in some aspects, other researchers suggested that any measure of fuzziness should be a measure of that lack of distinction between the fuzzy set and its complement [10], [25]. To evaluate uncertainty of answers to a given fuzzy query, Buckles and Petry [3] adapted De Luca and Termini's formula for measuring uncertainty in a fuzzy relational database. They provided a formula to compute the fuzzy entropy of a relation with respect to the query. However, measuring uncertainty is not suitable in determining more likely answers to a specific query, and we will discuss this argument in Section III. Therefore, the practical application of query evaluation to imprecise information is still lacking.

To consolidate and advance the state of the research in query processing for fuzzy relational databases, we provide another view of measuring the matching strength of an answer to a query by self-information, where self-information of a tuple is defined as the amount of information given by the tuple as an answer to the given query. That is, the matching strength of a tuple to a query can be

Manuscript received September 9, 1996; revised June 2, 1997 and November 8, 1997. This work was supported by the NSC Grant NSC 87-2213-E-032-001 of the Republic of China.

The authors are with the Department of Information Engineering, Tankang University, Taipei, Taiwan 251, R.O.C. (e-mail: chiang@cs.tku.edu.tw).

Publisher Item Identifier S 1094-6977(98)03898-X.