A fuzzy rule-based system consists of fuzzy if-then rules such as “If x1 is small and x2 is small than y is large”. The problem with existing fuzzy rule-based systems is that the size of the rule-base (number of rules) increases exponentially with the increase of the number of fuzzy sets involved in the rules. This exponential increase in size of the rule-base increases the search time and hence the problem solving time, and also the memory space required. In this paper a fuzzy rule-base compaction using genetic algorithm is proposed. The proposed approach consists of three phases: Knowledge acquisition phase, Encoding phase, and Compaction phase or Optimization phase. In knowledge acquisition phase information from various knowledge sources i.e. experts and machine learning methods is integrated into a single knowledge base. In encoding phase rule set and corresponding membership functions from different knowledge sources is encoded into a variable length string or chromosomes so that they can contribute to the genetic optimization approach. In optimization phase genetic algorithm that results in an optimal or nearly optimal set of fuzzy rules and membership functions from the initial set of rules and membership function is proposed.

**Keywords:** Fuzzy Rules, Genetic Algorithm, Optimal Rule-base, Rule-based Systems.

1. **INTRODUCTION**

Fuzzy logic improves classification and decision support of rule based systems by allowing the use of overlapping class definitions and interpretability of the used results by providing more insight the systems and decision making process. The automatic determination of fuzzy rules from data has been approached by several different techniques: neuro-fuzzy methods, fuzzy clustering in combination with GA. It is an open question as to what is the best way to extract rules from trained rule base in domains involving classification. The complete rule base for an expert system can be seen as an collection of different classification problem rules. The previous approaches were based on the exhaustive analysis of rule base have already been demonstrated to be intractable in that scale up factor increases exponentially with the increase in number of attributes.

The genetic algorithm (GA) is a combinatorial optimizer that is domain-independent; it is applicable to all functions that can be evaluated. Whereas hill-climbing and its relatives require domain-specific information (e.g., partial derivatives) to guide their searches, the genetic algorithm requires only two things: (1) a means of representing possible solutions and (2) an objective function evaluator—a function which maps a value from the domain of possible solutions to a scalar value. In the simplest terms, the genetic algorithm starts with an initial population of individuals each representing a point in the search space of a given function. Using an individual’s objective function as a measure of how “fit” that individual is within its environment, the genetic algorithm simulates nature’s survival of the fittest, essentially forcing the evolution of an optimal creature. This optimal creature is then the solution to the corresponding optimization problem.

2. **FUZZY KNOWLEDGE ACQUISITION PROCESS**

A fuzzy knowledge acquisition framework [1], shown in Fig 1, that accepts information from various fuzzy knowledge sources and converts them into a single knowledge base. Fuzzy rule set membership function and test objects including instances and historical records may be distributed among various sources. Knowledge from each site might be directly obtained by a group of human experts, using knowledge tools or delivered using machine learning methods may easily be translated into or represented by rules.

3. **RULE ENCODING PROCESS**

Since all the fuzzy rule set with their membership functions are obtained from different sources, they may differ in size, so a variable size Rule Representation [1] corresponding to a particular rule set with its membership function is used. Before encoding each fuzzy rule set is translated into a intermediary representation to preserve its semantic and syntactic constraints. This translation process involves the following steps:
3.1. Collect the features and possible values occurring in the condition parts of the fuzzy rule sets. All features gathered together comprise the global feature set.

3.2. Collect classes occurring in the conclusion parts of the fuzzy rule sets. All classes gathered together comprise global class set.

3.3. Translate each fuzzy rule into an intermediary representation that retains its essential syntax and semantics. If some features in the feature set are not used by the fuzzy rule, dummy tests are inserted into the condition part of the fuzzy rule. Each rule in the intermediary representation thus then composed of N feature tests and one class pattern. Where N is number of global features collected.

3.4. Concatenate all intermediary representations of rules to form the representation of rule set.

This intermediary representation can be well understood with the example explained below:

There are three species of iris flowers to be distinguished: Setosa, Versicolor, and Virginica. A class domain \( D_{\text{flower}} = \{ \text{Setosa}, \text{Versicolor}, \text{Virginica}\} \). Each rule is described by four features: Sepal Length(S.L.), Sepal Width(S.W.), Petal Length(P.L.), Petal Width(P.W.). Each feature has a domain given below:

\[
\begin{align*}
D_{\text{SL}} &= \{ \text{Short, Medium, Low} \} \\
D_{\text{SW}} &= \{ \text{Narrow, Medium, Wide} \} \\
D_{\text{PL}} &= \{ \text{Short, Medium, Low} \} \\
D_{\text{PW}} &= \{ \text{Narrow, Medium, Wide} \}
\end{align*}
\]

Assume a fuzzy rule set \( RS_q \) obtained from a fuzzy knowledge source has the following four rules:

\[
\begin{align*}
R_{q1} &: \text{IF (P.L.} = \text{Short) then Iris is Setosa;} \\
R_{q2} &: \text{IF (P.L.} = \text{Long) then Iris is Virginica;} \\
R_{q3} &: \text{IF (P.W.} = \text{Medium) then Iris is Versicolor;} \\
R_{q4} &: \text{IF (P.W.} = \text{Wide) then Iris is Virginica;}
\end{align*}
\]

After translation, the intermediary representation of the rules set would be constructed as follows:

\[
\begin{align*}
R'_{q1} &: \text{IF (S.L.} = \text{Short or Medium or Long) and IF (S.W.} = \text{Narrow or Medium or Wide) and IF (P.L.} = \text{Short) and IF (P.W.} = \text{Narrow or Medium or Wide) then Consq is Setosa.} \\
R'_{q2} &: \text{IF (S.L.} = \text{Short or Medium or Long) and IF (S.W.} = \text{Narrow or Medium or Wide) and IF (P.L.} = \text{Long) and IF (P.W.} = \text{Narrow or Medium or Wide) then Consq is Virginica.} \\
R'_{q3} &: \text{IF (S.L.} = \text{Short or Medium or Long) and IF (S.W.} = \text{Narrow or Medium or Wide) and IF (P.L.} = \text{Short or Medium or Long) and IF (P.W.} = \text{Medium) then Consq is Versicolor.} \\
R'_{q4} &: \text{IF (S.L.} = \text{Short or Medium or Long) and IF (S.W.} = \text{Narrow or Medium or Wide) and IF (P.L.} = \text{Short or Medium or Long) and IF (P.W.} = \text{Wide) then Consq is Virginica.}
\end{align*}
\]

The tests with underlines are Dummy tests. After translation each intermediary rule representation then consists of four feature sets and one consequent pattern. We then concatenate all intermediary rules to form an intermediary rule set \( RS'_q \). After each rule set has been translated into intermediary representation, each representation and membership functions are to be encoded. We are using the encoding structure:
Each chromosome is divided into two parts: RS'q and MFS_q. RS'q denotes an intermediary form of fuzzy rule set and MFS_q denotes the set of membership functions associated with RS'q. In Fig 2, R_{qi} represents the i-th rule in RS'q. T_{qij} is the j-th test in rule R_{qi}. a_{ij} denotes the j-th possible linguistic value of feature A_i. MF_{Ai} denotes the set of membership functions for feature A_i, and u_{aij} denotes the membership function of a_{ij}. Each feature test T_{qij} is an intermediary fuzzy rule set is then encoded into a fixed length binary substring s_{qi1}…s_{qim}, where m is the number of possible values for A_i. For example, assume the set of possible linguistic for feature A_i is \{a_{ij1}, a_{ij2}, a_{ij3}\}. Three bits are then used to represent this feature. The conclusion pattern in each rule is encoded as a bit string (1, 2…x) where x is the number of possible conclusions. When the rule concludes to l, then F_l is set as one and other as zero. N features tests and one conclusion pattern are then encoded and concatenated as a fixed length rule substring. All rules set substring are then concatenated to represent a variable length rule set string since each rule set differs in size.

For the above said example, the fuzzy rule R'q_1 is encoded as:

<table>
<thead>
<tr>
<th>S.L.</th>
<th>S.W.</th>
<th>P.L.</th>
<th>P.W.</th>
<th>Conseq</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>111</td>
<td>100</td>
<td>111</td>
<td>1001</td>
</tr>
</tbody>
</table>

Fig 3: Bit-string representation of R'q_1

Since feature S.L. in R'q_1 has three disjunctive test values, Short, Medium, Long, the test for S.L. is encoded as “111”. S.W. also has three disjunctive test values, Narrow, Medium, Wide, and thus is encoded as “111”. Similarly, P.W. is encoded as “111”. But P.L. has only one test value Short. It is thus encoded as “100”. Since the Consequent pattern has three possible values {Setosa, Versicolor, Virginica}, so the consequent pattern for Setosa is “100”.

All substrings of intermediary rules are then concatenated to represent the entire fuzzy rule set. The result is as shown in Fig 4.

We used two parameters to encode associated membership functions. Membership functions applied to a fuzzy rule set are then assumed to be isosceles-triangle functions as shown in Fig 5, where a_{ij} denotes the j-th linguistic value of feature A_i. u_{aij} denotes the membership function of a_{ij}. c_{ij} represents the center of abscissa of membership function u_{aij} and w_{ij} represents half the spread of membership function u_{aij}.

Fig 5: Membership Function for Feature A_i

Each membership function is represented as a pair (c, w). Thus all pairs of (c, w)’s for a certain feature are concatenated to represent its membership functions. MF_{qi} is then represented as a substring of c^{i1}_1, w^{i1}_1, ..., c^{imi}_mi, w^{imi}_mi where mi is the number of possible linguistic values of A_i (Fig 2). The entire set of membership functions MFS_q is encoded by concatenated substrings of MF_{A1}, MF_{A2}, ..., MF_{AN}. Since c and w are both numeric values, MFS_q is thus encoded as a fixed-length real-number than a bit string. Let the membership functions for each feature in RS_q are given in Fig 6.

Fig 6: Membership Functions for Iris Example.

According to the given encoding scheme MFS_q is represented as shown in Fig 7.

Fig 7: String Representation of MFS_q.
Since feature S.L. has three linguistic values, Short, Medium, Long, membership functions for S.L. are encoded as (5.2, 0.9, 6.1, 0.9, 7.0, 0.9) according to Fig 6. S.W. also has three linguistic values Narrow, Medium, Wide, and its associated membership functions are thus encoded as (2.6, 0.6, 3.2, 0.6, 3.8, 0.6). Similarly membership function for P.L. and P.W. are respectively, encoded as (2.4, 1.5, 3.9, 1.5, 5.4, 1.5) and (0.7, 0.6, 1.3, 0.6, 1.9, 0.6).

The fuzzy rule set RS\textsubscript{q} with associated membership function set MFS\textsubscript{q} is then encoded as shown in Fig 8.

![Fig. 8: String Representation of RS\textsubscript{q} with its Associated MFS\textsubscript{q}.](image)

According to this representation each chromosome consists of an intermediary fuzzy rule set and its associated membership functions. This representation allows the application of genetic operators to the multiple rules set with their membership functions at the same time.

4. RULE BASE OPTIMIZATION

4.1. Proposed Genetic Algorithm

After encoding of the rules in variable length chromosomes, a genetic algorithm is applied to optimize the number of rules (chromosomes) within the rule base. The proposed approach is shown in Fig 9. The various steps involved in the approach are explained below.

4.2. Initial Population

The genetic algorithm requires a population of feasible solutions to be initialized and updated during evaluation process of the proposed approach, the initial population of fuzzy rules comes from multiple knowledge sources. Each individual in the initial population consists of certain rules and membership functions from different knowledge sources.

4.3. Fitness and Selection

To develop a compact rule base from an initial population, the genetic algorithm selects parent fuzzy rules with high Fitness values for mating. An evaluation function is a set of test objects including the instances and historical records, which is then used to qualify the derived rule set. The performance of the derived rule sets their Fitness value is fed back to the genetic algorithm to continue. Now, how the solution space is searched to promote the quality fuzzy rules. Two factors are used in evaluating the derived fuzzy rule, Accuracy and Complexity which are defined below:

\[
\text{Accuracy (R}_i\text{)} = \frac{\text{Total Number of objects currently matched R}_i}{\text{Total number of objects}}
\]

\[
\text{Complexity} = \frac{\text{Number of rules into current population}}{\text{Total number of rules in the initial population}}.
\]

Then using these two functions we can determine the Fitness function as

\[
\text{Fitness (R}_i\text{)} = \frac{\text{Accuracy (R}_i\text{)}}{\text{[Complexity]}}
\]

Because our goal is to increase Accuracy and to decrease the Complexity of rule base. Here is a control parameter representing a trade-off between Accuracy and Complexity.

4.4. Genetic Operators

Genetic operators are very important to the success of a specific genetic algorithm. Two genetic Operators: Two-substring crossover and two-point mutation are used in the proposed approach.

4.4.1. Two Point Crossover

The two-substring crossover operator used, exchanges the substrings of the parents to generate offspring. It selects each crossover point with a probability of 1 / L-1, where L is the number of bits of the longest rule. The substring exchange may result in the change of both antecedent and consequent of the rule, so there are better chances of generation of better rules from the crossover operator. The three points chosen cprs, cpd and cpmf are chosen to accomplish this task, cprs is located in the rule-set part. And it may occur with in the rule string or at a rule boundary. The cprs crossover points need not be located at the same point positions for both parent chromosomes. The only requirement for cprs is that they must “match up syntactically” that means if one rule is cut at rule boundary the other parent must also be cut at a rule boundary. Similarly, if one parent is cut at a point p bits to left of a rule boundary, then the other parent must also be cut at a point p bits to left of some other rule boundary. cpd is always chosen at the boundary between the rule set and membership function set. cpmf is located in the membership function part and it may occur at any membership function center or at spreads. The cpmf crossover points for both parent chromosomes must however be same distances to the ends of both parent chromosomes. The crossover operator thus produces two offspring by exchanging the two substrings between cprs and cpd and cpmf and the end points of both parents. This is explained in Fig 10.
Fig. 9: Proposed Genetic Rule Base Compaction Approach
We have assumed that two parent rule set \( RS_1 \) and \( RS_2 \) respectively, contain \( k \) and \( h \) rules for classifying objects with two linguistic features \( (A_1, A_2) \). Features \( A_1 \) and \( A_2 \) both have two possible linguistic values. Two classes are to be determined. Also assume that \( RS_1 \) and \( RS_2 \) are encoded with their membership functions as shown in Fig 10(a).

\[
\begin{align*}
R_{i1} & : R_{i2} R_{i3} M_{F_{A1}} M_{F_{A2}} \\
RS_1 & + MFS_1: 101001...011001...1101001 5.708.699.111.60718.09 \\
R_{i1} & : R_{i2} R_{i3} M_{F_{A1}} M_{F_{A2}} \\
RS_2 & + MFS_2: 1101001...1001001...0101001 6.10970.099.130515.06
\end{align*}
\]

**Fig. 10(a): String Representation of \( RS_1 + MFS_1 \) and \( RS_2 + MFS_2 \)**

<table>
<thead>
<tr>
<th>( Q_{nov} )</th>
<th>( Q_{pt} )</th>
<th>( Q_{pt} )</th>
<th>( Q_{pt} )</th>
<th>( Q_{pt} )</th>
<th>( Q_{pt} )</th>
<th>( Q_{pt} )</th>
<th>( Q_{pt} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( RS_1 + MFS_1 )</td>
<td>( 101001...011001...1101001 )</td>
<td>( 5.708.699.111.60718.09 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( RS_2 + MFS_2 )</td>
<td>( 1101001...1001001...0101001 )</td>
<td>( 6.10970.099.130515.06 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After offspring fuzzy rule sets and their rule sets and their membership functions have been generated by two-substring crossover operation, the order of a newly generated fuzzy membership function and its neighbor may be destroyed. These fuzzy memberships thus need rearrangement according to their center values. An example is explained in Fig 11. Let after crossover the order of newly generated fuzzy membership function \((1.6, 0.5)\) and its neighbor \((1.5, 0.6)\) in the offspring O1 is destroyed. The crossover operator generates two offspring rule set O1 and O2. The pairs \((1.6, 0.5)\) and \((1.5, 0.6)\) for MFA in O1 are out of sequence. They are then rearranged in ascending order of membership function centers to \((1.5, 0.5) (1.6, 0.6)\).

### 4.4.2. Two-part Mutation

The two part mutation is used here to create a new fuzzy rule and a new fuzzy membership function from one chromosome. The mutation operator in the ruleset part and membership function part are different. Two part mutation operator at rule set part randomly changes bit values from a selected rule set, to escape the local-optimum traps. Two-part mutation operator applied to the membership function part creates new fuzzy membership function say n. Assume that c and w represent the center and spread of membership function. The center or spread of newly generated membership function will be \( c + e \) or \( s + e \) by the mutation operator. Mutation at the center of a fuzzy membership function may disrupt the order of the resulting fuzzy membership functions. These fuzzy membership functions then need rearrangement according to their center values, as shown in Fig 11.

**Fig. 11: Two-part Mutation Operation**

5. **Conclusion**

An optimization method for fuzzy rule base is proposed, that can be incorporated in fuzzy expert systems. Both
representative data and expert knowledge are included in designing fuzzy rules, which conventionally include expert knowledge. This approach does not require any human intervention during the optimization phase. The time required is thus dependent on the computer execution speed, but not on human experts, much time can thus be saved, since experts may be geographically dispersed, and their discussions are usually time consuming. The proposed approach is scalable to the increase in number of rule set. The increasing number of rule sets may increase the validity of the resulting knowledge base.

REFERENCES


