Chapter 5: Detailed Design Methodology

Algorithm

Step 1

Step 2

Step 5

Step 4

Step 3

Step 6

Step 7

Outcome
5.1 Chapter Overview

The Chapter 5 of the thesis discusses the necessities and the detailed design methodology of the generic framework in order to design intelligent system using evolutionary fuzzy approach. The discussed framework is based on the novel design of hybridization of Genetic Algorithm and Fuzzy Logic. The theory of MI in education domain has been considered as the application domain. The target is to suggest suitable career field to the student according to different types of intelligences he posses. MI constitutes 9 types of human intelligences, out of which five types have been considered in this application. The system is to be trained using the training data set, which consists of student’s score on 5 intelligences from set of Multiple Intelligence. To train the system, expert’s opinion about proposed future career field of the student has been considered. The developed intelligent system is designed in such a way that, it gets trained on expert’s opinion on this training data set and then independently predicts the student’s future career field.

The chapter presents the designed iterative procedure for developing generic framework using GA-Fuzzy Integration. The fuzzification and binary representation scheme for designing Genetic Algorithm is discussed in detail along with examples. GA is well known for providing two salient features such as the randomness and the evolution of the fittest genes. The Chapter discusses the significance of designing the specialized fitness evaluation process, which is a critical task, as the application itself lacks direct mathematical representation. In order to get the optimized results with minimum computation, different GA operators are applied over subsequent generation. The chapter further elaborates the need of genetic operators such as crossover, mutation, algebraic crossover. The research work provides design of specialized operator which sequentially combines different point crossover, mutation, arithmetic crossover and random operator. The chapter further elaborates the convergence criteria based upon minimum fitness and maximum GA generation, which defends how designed system provides optimum results. Thus, the chapter justifies the parameters for setting of GA to achieve the optimized outcome.

The outcome of a converged intelligent system would be a rule which predicts the proposed career field of the training data set, which would be equivalent to the expert’s choice. The system is then checked for the test data, for which the system
would independently predict the future career field without any dependence on the expert.

## 5.2 Steps for Designing Algorithm

The aim is to come up with a generic algorithm which is intuitive and computationally less intensive while automatically evolving a classifier from the training set. Calculation of score for total 5 types of intelligences is performed as a pre-processing step for obtaining the initial membership functions. This provides a good starting point for the member-ship functions because classifying the large number of data set may provide inconsistency in evolving rules and finally provides an accurate decision for suggesting an appropriate class for user.

The framework for generic architecture of the research work as represented in Chapter 4 of the thesis (refer Figure 4.3) is employed into following steps. Basically, the process of designing and handling Genetic Algorithm for solving the problem based on the non mathematical formula is very complex still it can be represented into following steps with the purpose of simplification.

### Step A: Fuzzification of Data

(a) Calculation of Truth Value or Degree of Membership ($\Phi$)

Truth value is calculated for each individual numerical score. To achieve this task, the function `CalculatePhi()` is designed, where $0 \leq \Phi \leq 1$.

(b) Definition of Fuzzy membership functions

Fuzzy membership functions are defined for linguistic representation. Every truth value is assigned for membership function. To achieve the stated task, the function `FuzzyMembership()` is designed.

### Step B: GA Rule Architecture

This step consists of several sub procedures which are explained below.

(a) Mapping of Intelligence into Rule

The research design provides a novel architecture to design rule using GA. In order to design the rule system, two design elements are considered as discussed below:

1. Number of intelligences considered in the domain of MI.
2. Number of proposed career fields as an outcome.

For the current research design, total five types of intelligences have been considered in the domain of MI. As an outcome of the system, three fields have been considered as the proposed career fields. For the decimal encoding scheme, considering 5 intelligence leads to number of bits for each profession to be 5 (1 bit * 5 intelligence). As total of 3 classes has been selected as outcome, each GA rule is of 15 bits (5 bits per intelligence * 3 bits per professions). The internal representation of rule can be further fragmented into equal intervals of 5 bits. First 5 integers represent the condition for 1st profession. Second 5 integers represent the condition for 2nd profession and third 5 integers from 21-30 represent 3rd profession. Each rule is represented as under.

\[ RB[15,1] = AED[1-5,1],AMD[6-10,1],AAD[11-15,1] \]

(b) Decimal to Binary Conversion

For GA encoding, 2 bits binary encoding scheme has been considered. The function \textit{Dec2Bin()} is designed to convert Decimal Rule to Binary. After the conversion each rule is represented as shown in following Equation.

\[ RB[30,1] = AEB[1-10,1],AMB[11-20,1],AB[21-30,1] \]

**Step C: Random Rule Generation**

In the initial stage of GA, 30 random rules are generated. The function \textit{GenerateRandomRule()} is designed to generate random rules. The following steps are performed on the generated rules.

(a) Penalty Evaluation for Each Profession with Training Data Set

Evolution and final optimization of the entire system is based upon meeting specific fitness criteria. In order to check the quality of population created in random rule, penalty of rule for each of the professions is calculated. To achieve this task, the function \textit{EvaluatePenalty()} is designed.

(b) Proposed Career Field by Rule

Based upon the penalty evaluation, GA rules predict the proposed career field of the students. The function \textit{ProposedCareerField()} is designed to achieve the prediction by GA rule.
(c) Fitness Evaluation

Proposed career field suggested by the rule has been checked with that of the Expert’s opinion to assign fitness of the rule to compare with the expert. The function \textit{CompareWithExpert()} is designed to evaluate fitness for rule.

\textbf{Step D: Sequential GA Operators & Convergence Check}

(a) Convergence Check

In order to identify whether optimized outcome is achieved, convergence check is required. Two criteria are considered for convergences which are enlisted as follows:

1. Number of generations, and
2. Fitness Value.

(b) Application of sequential GA Operators

In case of non-convergence, next population is generated by application of sequential GA operators as shown in following steps of D[1] to D[6] until optimized outcome is achieved. After each generation, Steps C.(a), C.(b) & C.(c) are performed.

- D[1]:GA Operation 1 - Mutation on 30 Bits $\rightarrow$ OperatorM30().
- D[2]:GA Operation 2 - Crossover on 10 Bits $\rightarrow$ OperatorC10().
- D[3]:GA Operation 3 - Crossover on 2 Bits $\rightarrow$ OperatorC2().
- D[4]:GA Operation 4 - Mutation on 2 Bits $\rightarrow$ OperatorM2().
- D[5]:GA Operation 5 - Arithmetic Crossover $\rightarrow$ OperatorAC().
- D[6]:GA Operation 6 - Infusion of Random Rules $\rightarrow$ OperatorR().

(c) Application on Test Data

Once a Converged solution is achieved, the value of outcome is applied to predict proposed career field. Test data is applied on converged GA rule as input and proposed career field is predicted by GA Rule.

The above mentioned procedure is summarized in the Figure 5.1(a) and the Figure 5.1(b) as under.

The pre-requisite of the system is represented using the Figure 5.1 (a). The research work is intended to provide a framework based on Genetic-Fuzzy integration in order to measure human intelligence for better future perspective. As discussed in the
Chapter 4, the aim of the designed system is to replace the role of the human expert in the domain of the theory of Multiple Intelligence (MI) in the education field. The designed system should accurately be able to predict the proposed career field of the student based upon the level of intelligence exhibited by the student in five different levels of MI, a task typically done by the human expert. The level of intelligence exhibited by the students in the field of Multiple Intelligence could be captured by providing a specific questionnaire. This questionnaire indicates the score of a student in dedicated intelligence based upon the response provided by the students to the question pertaining to that intelligence. In case of large volume of data elements, it is difficult to determine relative importance of different data elements. In order to solve this problem, fuzzy decision making approach is effectively utilized.

![Figure 5.1(a): Pre-Requisites for Design of Framework](image)

The designed iterative procedure for generic framework using Genetic-Fuzzy integration is presented in Figure 5.1(b).
As discussed in the Chapter 2, the absolute score as indicated by this questionnaire does not provide “degree of truth” which calls for application of Fuzzy Logic. The score of the students for each of the intelligence has separate membership functions defining particular score ranges. Each function maps the same score value in each of the intelligences to a truth value in the 0 to 1 range.
The following Equation represents calculation of fuzzified score from available score.

\[
\phi = \frac{\text{Achieved Score}}{\text{Maximum Possible Score}}
\]

The theory of Fuzzy Logic is useful in order to store and process domain specific linguistic knowledge. The application designed here is a typical application of classification which has numerical information available while Fuzzy Logic based system deals with linguistic knowledge. As discussed above, the score of questionnaires is basically numerical information. In order to convert, numerical information into linguistic knowledge, numerical information is classified into linguistic labels. Linguistic labels are associated to numerical score as per Table 5.1. For the current application, four linguistic labels are specified, such as:

1. Very High
2. High
3. Low
4. Mid

\(\phi\) represents truth value or degree of membership in the range of \([0, 1]\). In terms of the theory of Fuzzy Logic, this is called fuzzification of values. With the help of implementation of above Equation, fuzzification of numerical score is achieved.

**Table 5.1: Linguistic Representation of Intelligence Range using Truth Value**

<table>
<thead>
<tr>
<th>(\phi) (Truth Value)</th>
<th>Low</th>
<th>Mid</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0-0.2</td>
<td>Fairly</td>
<td>Slightly</td>
<td>Not</td>
<td>Not</td>
</tr>
<tr>
<td>0.2-0.5</td>
<td>Slightly</td>
<td>Fairly</td>
<td>Not</td>
<td>Not</td>
</tr>
<tr>
<td>0.5-0.8</td>
<td>Not</td>
<td>Slightly</td>
<td>Fairly</td>
<td>Not</td>
</tr>
<tr>
<td>0.8-1</td>
<td>Not</td>
<td>Not</td>
<td>Slightly</td>
<td>Fairly</td>
</tr>
</tbody>
</table>

Table 5.1 represents the “Essence” of Fuzzy Logic about interpreting the truth values applied to linguistic labels. For example, the truth value of 0.9 would lead to assignment in the intelligence range as “Not Low” & “Not Mid”. This means that the truth value corresponding to 0.9 is definitely “Not Low” and “Not Medium” on Intelligence. It is “Slightly High” & “Fairly Very High” on Intelligence range. Applying above this truth value belonging to Low & Mid Intelligence range is ruled out as; they both carry “Not” sign from the Table 5.1. This value could be a slight case of High Intelligence but the dominating indicator of this truth value is “Very
High” Intelligence range. This is due to the consideration that table assigns “Fairly” to
Very High Intelligence range. The difference between Fairly & Slightly is more of
qualitative. Slightly represents something which is “not more of” and fairly represents
something which is “Sufficiently more of”.

Here, in order to achieve fuzzification, membership functions have been designed. In
the Figure 5.2 the meaning of the expression Low, Mid, High and Very High are
represented by functions mapping the intelligence scale. Each point on that scale
possesses four “truth value” – corresponding to each of the four membership
functions. These membership functions are designed for all the five considered
intelligences. The application of this function on an absolute score of the students in
each of the intelligences provides the “graded value” for given intelligence.

![Fuzzy Membership Functions for Level of Intelligence](image)

**Figure 5.2: Fuzzy Membership Functions for Level of Intelligence**

### 5.3 GA System Architecture

The score of intelligence based on Fuzzy Logic based is calculated as explained
above. The resultant fuzzified values form the basis of the GA system design. As
stated earlier in the Chapter 2, there are various types of encoding schemes such as
binary encoding, tree encoding, permutation encoding and direct value encoding
available for designing solution of Genetic Algorithm. Due to several advantages of
binary encoding scheme, the current GA system has been designed as 2 bits binary
system. Rule architecture consists of the finite set of attributes with a finite number of
possible values can be easily represented using binary encoding. To convert student’s
intelligence level as achieved by “Low”, “Mid”, ”High”, and “Very High” level into 2
bits binary system, the following approach has been used. Table 5.2 represents the associated integers to the level of intelligence in order to map various levels of intelligence.

<table>
<thead>
<tr>
<th>Level of Intelligence</th>
<th>Assigned Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0</td>
</tr>
<tr>
<td>Mid</td>
<td>1</td>
</tr>
<tr>
<td>High</td>
<td>2</td>
</tr>
<tr>
<td>Very High</td>
<td>3</td>
</tr>
</tbody>
</table>

As discussed in the Chapter 4, five different kinds of intelligences are considered. These are enlisted as under:

1. Interpersonal
2. Intrapersonal
3. Logical
4. Musical/Visual
5. Verbal

The overall aim of the designed system is to replace the role of an expert by predicting student’s proposed career field based on the level of intelligence the student shows on the above different type of intelligences. In order to have prediction of career fields for the students, following fields have been considered for the application:

1. Engineering,
2. Entrepreneurship, and
3. Art.

It is well known that every human being acquire different types of intelligences in different measures. However we may include many other levels of measure. Here, the task is to decide appropriate career field using combination of intelligence from the set of intelligences. In order to decide appropriate career field for the student, five different types of intelligences from the set of MI are required to be measured. The rules are structured using combination of different intelligences and their suitable
levels of measures for identifying suitable career fields. For designing these rules many academicians, field experts and industrial experts have been consulted.

Main architecture of GA rule considers five intelligences of a student for each of the future career fields. This can be typically shown using Figure 5.3 as under.

![Figure 5.3: Structure of GA Rule](image)

Where $A_E$ represents the students’ level in Logical Intelligence for the category, if the student’s proposed career field is “Engineering”. Similarly $B_E, C_E, D_E, E_E$ represent the student’s level in Interpersonal, Intrapersonal, Musical & Verbal intelligence, if the proposed career field is “Engineering”. Similarly $A_M, B_M, C_M, D_M, E_M$ represent the score for those intelligences levels if the field is “Entrepreneurship” & $A_A, B_A, C_A, D_A, E_A$ represent the score for those intelligence level if the field is “Art”.

For example, Table 5.3 shows expert’s prediction of suitable career fields of the students according to the type of intelligence and levels of intelligence.

<table>
<thead>
<tr>
<th>Type of Intelligence</th>
<th>Level of Intelligence</th>
<th>Career Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>Very High</td>
<td>Engineering</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Interpersonal</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Musical/Visual</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Verbal</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td>Mid</td>
<td>Entrepreneurship</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td>Mid</td>
<td></td>
</tr>
<tr>
<td>Interpersonal</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Musical/Visual</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Verbal</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td>Low</td>
<td>Art</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td>Very High</td>
<td></td>
</tr>
<tr>
<td>Interpersonal</td>
<td>Mid</td>
<td></td>
</tr>
<tr>
<td>Musical/Visual</td>
<td>Very High</td>
<td></td>
</tr>
<tr>
<td>Verbal</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

Then applying Table 5.2 on Table 5.3 would lead to a rule as shown in Figure 5.4.

![Figure 5.4: Typical Rule after Mapping Integer Value to Intelligence Level](image)
5.3.1 Binary Rule Representation for GA

2 bits binary system is considered as the encoding scheme. 2 bits binary system completely satisfies minimum and maximum integer values i.e. 0 & 3. 2 bits binary system is also economical considering computation power requirement. The integer, assigned based on the fuzzy intelligence score, is converted to binary system, as shown in the Table 5.4.

Table 5.4: Conversion of Decimal Integer To 2 Bits Binary System

<table>
<thead>
<tr>
<th>Decimal Integer</th>
<th>2 Bit Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
</tr>
</tbody>
</table>

Applying the above conversion to Figure 5.5 would lead to binary representation of GA rule as shown below.

```
0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 0
```

Figure 5.5: Binary Representation of GA Rule

5.3.2 Training Data Set

Training data set is of the prime importance in the system design because the design first gets “Trained” on this data and then makes prediction on its own. Training data is selected such that it captures entire search space. Nine training data sets are considered. The training data set is also encoded and embedded within the underlying GA system and is only modified inside the code. This is due to the consideration that, it has the biggest impact on the development and prediction capability of the evolved system. Typical training data considered as shown in Table 5.5.
**Table 5.5: Training Data Set Designed for Prediction**

<table>
<thead>
<tr>
<th>Training Data Number</th>
<th>Student’s Score in Intelligence</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logical</td>
<td>Intra personal</td>
<td>Inter personal</td>
<td>Musical/Visual</td>
<td>Verbal</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>9</td>
<td>5</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>12</td>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>4</td>
<td>15</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>8</td>
<td>17</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>10</td>
<td>19</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>10</td>
<td>7</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>13</td>
<td>4</td>
<td>19</td>
<td>5</td>
</tr>
</tbody>
</table>

Expert’s opinion is taken on the above training data set for the proposed career fields of the students. Here, as explained above, three career fields have been considered, i.e. Engineering, Entrepreneurship & Art. Expert based upon his experience and knowledge selects one career field for each of the student. Table 5.6 captures the expert’s decision about the proposed career field for the above training data set.

**Table 5.6: Expert’s Decision about Students’ Score**

<table>
<thead>
<tr>
<th>Training Data Number</th>
<th>Student’s Score in Intelligence</th>
<th>Expert’s Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logical</td>
<td>Intra personal</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>13</td>
</tr>
</tbody>
</table>

The above mentioned score does not represent the “truth value” of the intelligence possessed by the students hence, fuzzy membership functions as defined in the Figure. 5.1 have been applied. Fuzzy Intelligence score is calculated. Maximum possible score is considered as 20.

For Example, Logical intelligence of Training Data number 1 is 15. Fuzzy intelligence score is calculated as = 15/20 = 0.75. Now applying Table 5.1 on this
score would indicate intelligence level of “Fairly High” and hence integer assigned is 2 as per the Table 5.2. Following the same methodology would lead to Table 5.7.

Table 5.7: Representation of Students Score After Fuzzification

<table>
<thead>
<tr>
<th>Training Data Number</th>
<th>Student’s Score in Intelligence</th>
<th>Expert’s Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logical</td>
<td>Intrapersonal</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Simulating architecture of rules as shown in the above table, the first training data can be represented as shown in Figure 5.6.

Figure 5.6: Example of Rule for Training Data

5.3.3 GA Random Rules Generation

As explained in the Chapter 4, randomness is one of the most prominent features of GA. In the initial phase of the system, 30 random rules are generated. The final outcome of the entire GA system is to train these rules on the training data set, via sequential generation by using GA operator. The most fit, rule evolves in the end, would have capability to predict the proposed career field of a student, thus replacing the role of an expert. For example, if a random rule is generated, at first iteration, it can be represented in Figure 5.7 (a).

Figure 5.7(a): Example of Random Rule Generated at First Iteration

As per algorithm design, converting the bit strings into 2 bits binary system would lead to Figure 5.6 (b).

Figure 5.7(b): Conversion of Random Rule into 2 bits binary system
5.3.4 Fitness Evaluation

The fitness evaluation is of the most important aspect of system design because evolution and final optimization of the entire system is based upon meeting with the specific fitness criteria. Also the selected problem pertaining to non mathematical problem domain, the fitness evaluation is a challenging task. The fitness evaluation for each of the rules is carried out as per the following process:

As discussed above GA rules are constructed in such a way that 5 integers represent intelligence required for a specific career field. Now, fitness for each of the professions for the GA rule is calculated against that of the training data set. For example, intelligence level for “Engineering” by the GA rule can be represented as Figure 5.8(a).

![Figure 5.8(a): Representation of GA rule for Intelligence Level for Engineer](image)

Training data set 1 is represented as shown in Figure 5.8(b).

![Figure 5.8(b): Representation of Training Data Set 1](image)

Fitness of the GA rule for each of the profession is calculated by subtracting the intelligence value of the GA rule to that of the training data set. For example, Table 5.8 represents calculation of fitness by comparing values of the GA rule and the training data. Here, data values are used from Figure 5.8(a) and Figure 5.8 (b).

<table>
<thead>
<tr>
<th>Type of Intelligence</th>
<th>GA Rule for Engineer</th>
<th>Training Data</th>
<th>Difference between GA rule &amp; Training Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>0</td>
<td>2</td>
<td>0 – 2 = -2</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td>2</td>
<td>1</td>
<td>2 – 1 = 1</td>
</tr>
<tr>
<td>Interpersonal</td>
<td>3</td>
<td>1</td>
<td>3 – 1 = 2</td>
</tr>
<tr>
<td>Musical/Visual</td>
<td>0</td>
<td>1</td>
<td>0 – 1 = -1</td>
</tr>
<tr>
<td>Verbal</td>
<td>0</td>
<td>1</td>
<td>0 – 1 = -1</td>
</tr>
</tbody>
</table>

Table 5.8: Example of Calculation of Fitness
The physical interpretation of the above mentioned process is summarized as under.

If a student’s proposed career field is “Engineering”, for logical intelligence, the GA rule suggests having a value of 0. However, for the training data set this value is 2. Difference between these two values is (-2).

Now penalty is applied if the difference between these two numbers is positive, meaning if the training data set has less value compared to the GA rule (having a positive value after subtraction). In such case, the penalty of the same has been applied for that intelligence value. If the difference is negative (training data set having more value than required intelligence value), then no penalty is applied. In general, the rules for penalty are defined as follows:

1. If Difference between GA Rule & Training Data ≤ 0 then
   Assigned Penalty = 0.
2. If Difference between GA Rule & Training Data > 0 then
   Assigned Penalty = Difference.

As for the condition, the training data set has more than required value for this intelligence (as defined by having a negative value); the penalty for the same has been considered to be 0. Taking an example for Intrapersonal intelligence, the GA rule suggests having a value of 2. However, for the training data set this value is 1. For this case, the minimum requirement is not met. For this case, the difference in intelligence is 1 and hence penalty applied is 1.

Taking another example for Musical intelligence, the GA rule suggests having a value of 0. However, for the training data set, this value is 1. Difference between these two values is -1. For this case, minimum penalty criterion is met. Hence, penalty of 0 is applied for this intelligence.
Table 5.9: Calculation of Penalty for the Complete Rule

<table>
<thead>
<tr>
<th>Type of Intelligence</th>
<th>Profession</th>
<th>GA Rule</th>
<th>Training Data</th>
<th>Diff. between GA rule &amp; Training Data</th>
<th>Assigned Penalty ($p_i$)</th>
<th>Total Penalty for a given Profession ($\sum p_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpersonal</td>
<td>Engineering</td>
<td>0</td>
<td>2</td>
<td>$0 - 2 = -2$</td>
<td>0</td>
<td>$0 + 1 + 2 + 0 + 0 = 3$</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td></td>
<td>2</td>
<td>1</td>
<td>$2 - 1 = 1$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td></td>
<td>3</td>
<td>1</td>
<td>$3 - 1 = 2$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Musical /Visual</td>
<td></td>
<td>0</td>
<td>1</td>
<td>$0 - 1 = -1$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Verbal</td>
<td></td>
<td>0</td>
<td>1</td>
<td>$0 - 1 = -1$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Interpersonal</td>
<td>Entrepreneurship</td>
<td>2</td>
<td>2</td>
<td>$2 - 2 = 0$</td>
<td>0</td>
<td>$0 + 0 + 0 + 0 + 0 = 0$</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td></td>
<td>1</td>
<td>1</td>
<td>$1 - 1 = 0$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td></td>
<td>1</td>
<td>1</td>
<td>$1 - 1 = 0$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Musical /Visual</td>
<td></td>
<td>0</td>
<td>1</td>
<td>$0 - 1 = -1$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Verbal</td>
<td></td>
<td>0</td>
<td>1</td>
<td>$0 - 1 = -1$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Interpersonal</td>
<td>Art</td>
<td>1</td>
<td>2</td>
<td>$1 - 2 = -1$</td>
<td>0</td>
<td>$0 + 2 + 0 + 2 + 0 = 5$</td>
</tr>
<tr>
<td>Intrapersonal</td>
<td></td>
<td>3</td>
<td>1</td>
<td>$3 - 1 = 2$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td></td>
<td>0</td>
<td>1</td>
<td>$0 - 1 = -1$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Musical /Visual</td>
<td></td>
<td>3</td>
<td>1</td>
<td>$3 - 1 = 2$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Verbal</td>
<td></td>
<td>2</td>
<td>1</td>
<td>$2 - 1 = 1$</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Based upon the above total calculated penalty, the GA rules predict the proposed career field of the student. To achieve this task, it considers Minimum value of the penalty for all three professions.

For example, Table 5.9 calculates total penalty profession wise: 3, 0, 5 for Engineering, Entrepreneurship and Art respectively. As per algorithm, minimum penalty from these groups is calculated i.e. $\text{Min}(3, 0, 5) = 0$ which is representing Entrepreneurship. Hence, the GA rule would predict the proposed career field as Entrepreneurship.

Now, the prediction done by the GA rule based upon the minimum penalty is then compared with that of the expert’s decision about proposed career field. If the
prediction by the GA rule matches that of the expert’s decision, then fitness penalty of the GA rule is considered to be 0. If this does not match then fitness penalty of 1 is to be applied. By summarizing the stated process, equation 5.4 can be designed which shown using following Equation as under:

If Prediction by GA rule = Prediction by Expert then Fitness Penalty = 0
else
fitness Penalty = 1

The above procedure is repeated for each of the GA rule for all the 9 training data set. The final fitness penalty of the GA rule for all the training data set is the summation of fitness penalty for each training data set. For example, if for all the 9 training data sets, the GA rules prediction does not match with that of the expert’s decision for all the cases, then final fitness penalty of 9 would be applied.

As discussed above the total 30 GA rules are generated and hence, this procedure is carried out for all the 30 rules.

Now, it can very well happen that, while evaluating the fitness of each rule by subtracting the intelligence value of the GA rule with that of the training data set, penalty for multiple profession may end up having same minimum value. For example, in the Table 5.9, the penalty for both the professions: “Engineering” and “Entrepreneurship” have same value of 1. In this case, GA would select both of these fields, as possible career fields. Now, if any of the career fields predicted by GA matches that of the expert’s decision, then fitness penalty of 0.5 would be applied as final fitness penalty. Main consideration of including this inside the system is that, even though GA is predicting a result similar to that of the expert’s (by predicting at least one right profession); it also provides an alternative career field, which is NOT suggested by an expert. Hence, penalty assigned is 0.5 rather than 0 for the completely matching prediction as that of the expert. The rules for fitness penalty are summarized in Table 5.10.
Table 5.10: Rules for Fitness Penalty

<table>
<thead>
<tr>
<th>Number of Career Field Predicted by System</th>
<th>Criteria with Matching that of Expert’s choice</th>
<th>Assigned Fitness Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Complete Match</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>One of the field match with expert’s decision</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>None of the field match with expert’s decision</td>
<td>1</td>
</tr>
</tbody>
</table>

All the 30 GA rules are then sorted as per their ascending fitness penalty to generate new generations of GA.

5.3.5 Sequential Generations via GA

To subsequent generation via applying specific GA operators, 30 rules of previous generations are used in a specific way as shown below.

- Rules 1 to 10 (first 10 Fittest rules) are included in the next generation as these are the rules 1 to 10 (First 10 fittest rules) and the rules 21 to 30 (Last 10 fittest rules) act as a parent to generate further 20 rules by application of Crossover & Mutation operators as described in sections below.

5.3.5.1 GA Operator 1 – Mutation on 30 Bits

The first GA operator considered application of Mutation operator is applied on 30 bits of the GA rule. Parents for this operator are the rules 1 to 10 (first 10 fittest rules) and rules 21 to 30 (Last 10 fittest rules) of the previous generation after arranging them as per the ascending order of fitness.

There are 2 main considerations for the application of this specific operator which can be discussed as under.

1. Mutation operator applied over the fittest rules & the least fit rules have potential to create an off-spring which is genetically superior due to inherent mutation operator characteristic.

2. Selection of 30 bits to be considered for mutation operation, is due to the fact that 30 bits in the GA rule represent each career field (5 bits for Intelligence) * 2 bits (for binary representation) * 3 (bits for profession)). Mutation of this string, has higher probability to convert the High penalty profession into the Low penalty in the next generation (due to inherent characteristic of mutation
Chapter 5: Detailed Design Methodology

A Genetic-Fuzzy Approach to Measure Multiple Intelligence

operator) and hence higher chances of having a better fitness overall. A typical example of this operator can be described as shown in Figure 5.9 (a), Figure 5.9 (b), Figure 5.9(c) and Figure 5.9(d). Parent string for mutation operator is shown in Figure 5.9 (a).

**Parent 1**

```
0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 1 0
```

*Figure 5.9(a): Binary Bit String for Parent 1*

Mutation Over Parent 1 as shown in Figure 5.9 (b):

**Child 1**

```
1 1 0 1 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1
```

*Figure 5.9(b): Binary Bit String after applying GA Operator on Parent 1*

**Parent 2**

```
0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 0
```

*Figure 5.9(c): Binary Bit String of Parent 2*

**Child 2**

```
1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 1
```

*Figure 5.9(d): Binary Bit String after applying Mutation on Parent 2*

After generating another 20 rules by the above operation, the fitness evaluation is made once again as already described, for the entire training data set. After sorting again, the process of evaluating fitness of rules as per the ascending penalty, acts as a basis for the next generation of GA.

5.3.5.2 GA Operator 2 – Crossover on 10 Bits

The second GA operator considered application of crossover operator applied on 30 bits of the GA rule. Parents for this operator are the rules 1 to 10 (first 10 fittest rules) and the rules 21 to 30 (Last 10 fittest rules) of the previous generation after arranging them as per the ascending order of fitness. There are 2 main considerations for the application for this specific operator as explained below:

1. Crossover operator applied over the fittest rules & the least fit rules have potential to create an offspring which is genetically superior due to
consideration that replacing the genes of the poor rules by that of the fitter rules could result into superior offspring.

2. Selection of 10 bit to be considered for crossover operation, is due to the fact that, 10 bits in the GA rule, represents each Career field (5 Intelligence * 2 bit binary). Crossover of this string has higher probability to convert the high penalty assigned to specific profession to have low penalty in the next generation (due to replacing inferior genes with superior by Crossover as explained above) and hence higher chances of having a better fitness overall. Due to this consideration of handling each profession at a time, this operator repeats itself as explained below.

A typical example of this operator is described in the Figure 5.10. For the first iteration, this operator is applied on the first 10 bits of each parent as shown in Figure 5.10 (a), Figure 5.10 (b), Figure 5.10 (c) and Figure 5.10 (d).

**Parent 1**

```
1 1 0 1 0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1
```

**Parent 2**

```
1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0
```

**Figure 5.10(a): Operation on First 10 Bits of Parent 1**

**Figure 5.10(b): Operation on First 10 Bits of Parent 2**

Application of the operator would lead to creation of following children as shown in the Figure 5.10 (c) and Figure 5.10 (d).

**Child 1**

```
1 0 0 1 0 1 1 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1
```

**Figure 5.10(c): Result of Operator Applied on Parent 1**

**Child 2**

```
1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 1
```

**Figure 5.10(d): Result of Operator Applied on Parent 2**

After generating another 20 rules by the above operation, the fitness evaluation is performed once again as already described for the entire training data set. After sorting again the process of fitness evaluation of rules is made as per the ascending penalty which acts as basis for the next generation of GA. As discussed previously,
this operator is applied iteratively for each generation. Hence, for the next generation of rules, this operator is applied on the next 10 bits (11 to 20) of each parent as shown in the Figure 5.11 (a).

Parent 1

![Figure 5.11(a): Operators for next generation of Parents using 11 to 20 bits](image)

Parent 2

![Figure 5.11(b): Operators for next generation of Parents using 11 to 20 bits](image)

An application of the operator would lead to creation of child1 as shown in the Figure 5.11 (c).

Child 1

![Figure 5.11(c): Result of operators for Next Generation of Parents](image)

Child 2

![Figure 5.11(d): Result of Operators for Next Generation of Parents](image)

After rearranging the rules by ascending penalty for the next generation of rules, this operator is applied on the next 10 bits (21 to 30) of each parent as shown in the Figure 5.12 (a), Figure 5.12 (b), Figure 5.12 (c) and Figure 5.12(d).

Parent 1

![Figure 5.12(a): Operators for next generation of Parents using 21 to 30 bits](image)

Parent 2

![Figure 5.12(b): Operators for next generation of Parents using 21 to 30 bits](image)

Application of the operator would lead to creation of following Child1 and Child 2 as shown in the Figure 5.12 (c) and Figure 5.12 (d).
After rearranging the rules by the ascending penalty, the GA operator 3 – Crossover on 2 Bits is applied as shown in the following section.

5.3.5.3 GA Operator 3 – Crossover on 2 Bits

The third GA operator considered is Crossover applied on 2 bits of the GA rule. Parents for this operator are the rules 1 to 10 (first 10 fittest rules) and the rules 21 to 30 (Last 10 fittest rules) of the previous generation after arranging them as per the ascending order of fitness. There are two main considerations for the application for this specific operator which are presented as follows:

1. Crossover operator applied over the most fit rules & the least fit rules have potential to create an offspring which is genetically superior due to consideration that replacing the genes of the poor rules by that of the fitter rules could have superior off-springs.

2. Selection of 2 bits to be considered for crossover operation is due to the fact that 2 bits in the GA rule represent a specific intelligence in the GA rules. Crossover of 2 bits can have higher probability to convert the high penalty assigned to a specific intelligence in a specific profession to have low penalty in the next generation (due to replacing inferior genes with superior by Crossover as explained above) and hence higher chances of having a better fitness overall. Due to this consideration of handling each profession at a time, this operator repeats itself as explained below.

A typical example of this operator is described as under. For the first iteration, this operator is applied on the first 10 bits of each parent as shown in the Figure 5.13(a) and the Figure 5.13(b).
Figure 5.13(a): 2 Bit Crossover Operator on Parent 1

Figure 5.13(b): 2 Bit Crossover Operator on Parent 2

Application of the operator would lead to creation of following child as shown in the Figure 5.13(c) and Figure 5.13(d).

Figure 5.13(c): Result of 2 Bit Crossover on Parent 1

Figure 5.13(d): Result of 2 Bit Crossover on Parent 2

As discussed previously, this operator is applied iteratively to each generation. After rearranging the rules by the ascending penalty for the next generation of rules, this operator is applied on the next 10 bits (3 & 4) of each parent as shown in the Figure 5.14 (a), Figure 5.14 (b), Figure 5.14 (c) and Figure 5.14 (d).

Figure 5.14 (a): 2 Bit Crossover Operator on next 2 bits of Parent 1

Figure 5.14 (b): 2 Bit Crossover Operator on 3 & 4 of Parent 2

Application of the operator would lead to creation of following child as shown in the Figure 5.14 (c).

Figure 5.14(c): Result of 2 Bit Crossover Operator on 3 & 4 bits of Parent 1
Child 2

```
1 0 0 1 0 1 1 0 1 0 0 1 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1
```

Figure 5.14(d): Result of 2 Bit Crossover Operator on 3 & 4 bits of Parent 2

After rearranging the rules by the ascending penalty for the next generation of rules, this operator is applied on the next 2 bits (5 & 6) of each parent as shown in the Figure 5.15(a) and Figure 5.15 (b).

Parent 1

```
1 1 0 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1
```

Figure 5.15(a): 2 Bit Crossover Operator on 5 & 6 bits of Parent 1

Parent 2

```
1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0 1 0 1
```

Figure 5.15(b): 2 Bit Crossover Operator on 5 & 6 bits of Parent 1

Application of the operator would lead to creation of following children as shown in the Figure 5.15(c) and Figure 5.15(d).

Child 1

```
1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 1
```

Figure 5.15(c): Result of 2 Bit Crossover Operator on 5 & 6 bits of Parent 1

Child 2

```
1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 1
```

Figure 5.15(d): Result of 2 Bit Crossover Operator on 5 & 6 bits of Parent 2

The above mentioned process is followed till the end of the GA rule. After the final application of this operator on the last digit of the GA rule, the rules are once again rearranged as per the ascending penalty and then GA Operator 4 – Mutation on 2 Bits is applied as described below is applied.

5.3.5.4 GA Operator 4 – Mutation on 2 Bits

The fourth GA operator considered is Mutation applied on 2 bits of the GA rule. Parents for this operator are the rules 1 to 10 (first 10 fittest rules) and the rules 21 to 30 (Last 10 fittest rules) of the previous generation after arranging them as per
ascending order of fitness. There are 2 main considerations for the application for this specific operator which are described as follows:

1. Mutation operator applied over the fittest rules and the least fit rules have potential to create an off-spring which is genetically superior due to consideration that flipping the genes of the poor rules could have superior off-springs.

2. Selection of 2 bits to be considered for mutation operation is due to the fact that 2 bits in the GA rule represent a specific intelligence in the GA rules. Mutation of 2 bits, can have higher probability to convert the high penalty assigned to a specific intelligence in a specific profession to have low penalty in the next generation (due to flipping the property of inferior genes with superior by Mutation as explained above) and hence higher chances of having a better fitness overall. Due to this consideration of handling each profession at a time, this operator repeats itself as explained below.

A typical example of this operator is described as shown in the Figure 5.16 (a), Figure 5.16 (b), Figure 5.16 (c) and Figure 5.16 (d).

**Parent 1**

| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

*Figure 5.16(a): First Parent String before Mutation Operation*

**Parent 2**

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

*Figure 5.16(b): Second Parent String before Mutation Operation*

Application of the operator would lead to creation of following child as shown in the Figure 5.16(c).

**Child 1**

| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

*Figure 5.16(c): 2 Bits Mutation Operation on Parent 1*

**Child 2**

| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

*Figure 5.16(d): 2 Bits Mutation Operation on Parent 2*
As discussed previously, this operator is applied iteratively to each generation. After rearranging the rules by the ascending penalty for the next generation of rules, this operator is applied on next 10 bits (3 & 4) of each parent as shown in the following Figures: Figure 5.17 (a), Figure 5.17(b), Figure 5.17(c), and Figure 5.17(d).

Parent 1

```
1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1
```

Figure 5.17(a): Next Generation Parent 1 after ascending Penalty

Parent 2

```
1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 1
```

Figure 5.17(b): Next Generation Parent 2 after ascending Penalty

Application of the operator would lead to creation of the following child as shown in the Figure 5.17 (c).

Child 1

```
1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 0 0 1
```

Figure 5.17(c): Result of 2 bits mutation on Parent 1 of Next Generation

Child 2

```
1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1
```

Figure 5.17(d): Result of 2 bits mutation on Parent 2 of Next Generation

After rearranging the rules by the ascending penalty for the next generation of rules, this operator is applied on next 2 bits (5 & 6) of each parent as shown in the following Figures: Figure 5.18 (a) and Figure 5.18 (b).

Parent 1

```
1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1
```

Figure 5.18(a): Next Generation Parent 1 after ascending Penalty

Parent 2

```
1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1
```

Figure 5.18(b): Next Generation Parent 2 after ascending Penalty
Application of the operator would lead to creation of the following children as shown in the Figure 5.18(c), and the Figure 5.18(d).

**Child 1**

\[
\begin{array}{cccccccccccccccc}
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0
\end{array}
\]

**Figure 5.18(c): Result of 2 bits Mutation on Parent 1 of Next Generation**

**Child 2**

\[
\begin{array}{cccccccccccccccc}
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}
\]

**Figure 5.18(d): Result of 2 bits Mutation on Parent 2 of Next Generation**

The above mentioned process is followed till the end of the GA rule. After the final application of this operator on the last digit of the GA rule, the rules are once again rearranged as per the ascending penalty and then the GA Operator 5 – Arithmetic Crossover is applied as described below.

**5.3.5.5 GA Operator 5 – Arithmetic Crossover**

The Fifth GA operator considered is Arithmetic Crossover. Parents for this operator are the rules 1 to 10 (first 10 fittest rules) and the rules 21 to 30 (Last 10 fittest rules) of the previous generation after arranging them as per the ascending order of fitness. The main consideration of using this sub-genre of crossover operator is that changing the values of \( \alpha \) & \( \beta \) presents a scope of sensitivity study to be carried out if it is found a significant impact on the final penalty outcome.

A typical example of this operator is described as shown in the Figure 5.19 (a), the Figure 5.19 (b), the Figure 5.19(c) and the Figure 5.19(d).

\[
x' = \alpha x + \beta y
\]
\[
y' = \beta x + \alpha y
\]

where \( 0 \geq \alpha, \beta \leq 1 \) & \( \alpha + \beta = 1 \).

At first this rule is applied on 10 bits as following. The considered value of \( \alpha = 0.1, \beta = 0.9 \).
A Genetic-Fuzzy Approach to Measure Multiple Intelligence

Parent 1

\[
\begin{array}{cccccccccccc}
X_1 & X_2 & \ldots & \ldots & \ldots & X_{10} \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Figure 5.19(a): Parent 1 before applying Arithmetic Crossover Operator

Parent 2

\[
\begin{array}{cccccccccccc}
Y_1 & Y_2 & \ldots & \ldots & \ldots & Y_{10} \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
\]

Figure 5.19(b): Parent 2 before applying Arithmetic Crossover Operator

\[
x'_1 = 0.1 \times 1 + 0.9 \times 0 = 0.1 \\
y'_1 = 0.9 \times 1 + 0.1 \times 0 = 0.9
\]

However, as discussed above, the 2 bits GA system cannot recognize decimal number such as 0.5. The same has thus been rounded off to the nearest integer of 1, as per the following rule:

If \( x'_i \) or \( y'_i \) \( \leq 0.5 \) Then \( x'_i \) or \( y'_i = 1 \)

If \( x'_i \) or \( y'_i > 0.5 \) Then \( x'_i \) or \( y'_i = 0 \)

Calculating offspring as per the above rule would lead to creation of the following children as shown in the Figure 5.19(c) and Figure 5.19(d).

Child 1

\[
\begin{array}{cccccccccccc}
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Figure 5.19(c): Child 1 after applying Arithmetic Crossover

Child 2

\[
\begin{array}{cccccccccccc}
1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
\]

Figure 5.19(d): Child 2 after applying Arithmetic Crossover

After rearranging the rules by the ascending penalty, for the next generation of rules this operator is applied on the next 10 bits of each parent as shown in the Figure 5.19(e), Figure 5.19(f), Figure 5.19(g) and Figure 5.19(h).
After rearranging the rules by the ascending penalty, for the next generation of rules this operator is similarly applied on the next 10 bits of each parent.

**5.3.5.6 GA Operator 6 – Random Rule infusion**

Randomness plays a critical role in the GA system design. But the inherent characteristic of randomness is that, many times it is not possible to achieve convergence due to randomness. The solution of this problem is to add specific rules which help system to converge [154]. Thus, in the case where a system is having non-convergence state, it may take help of infusion of random rules in order to achieve convergence. To capture the same issue, 10 random rules are infused into the system, which replaces the worst 10 rules of the previous generation. After this infusion, the rules are once again rearranged as per the ascending penalty.

**5.4 Sequence of Application of Operators**

The above mentioned operators are sequentially applied over each generation in order to imitate the prediction of the expert’s decision about the student’s proposed career field. For each generation, the sequence of application of operators is presented as shown in the Figure 5.20.
Figure 5.20: Sequence of Application of Operators over Each Generation

5.4.1 Convergence Criteria

The convergence criterion is one of the most important parameters for system design. Too lenient convergence criteria would lead to lower computational requirements, but may result in non-accurate GA system design, which may not meet the desired aim of replacing the role of the human expert. On the other hand, too stringent convergence criteria would lead to an accurate GA system which efficiently replaces the role of human expert, but may require too high computation requirement for the system to converge. Considering both of these aspects, the following criteria have been considered for acceptable GA system design.

1. Minimum Fitness of 0.5.
2. Maximum number of generation of 50.

The minimum criterion for fitness is considered to be 0.5. This is due to the fact that a penalty of 0.5 would make sure that the GA system is correctly predicting future career fields for all the 9 training data sets matching that with the expert’s decision, “Except” for one data set where it also predicts an alternative career field. The penalty at each generation, after sequential application of the GA operators as described above, is calculated and in case it is found to be 0.5 or less the system is found to be
an acceptable fitness and the process is terminated based upon the minimum fitness criteria.

The maximum criterion for number of generation is considered to 50. This is due to the fact that, if the GA system fails to achieve the required minimum fitness of 0.5 after 50 generations, it has already consumed fair amount of computational power and the criteria is set “NOT” to consume exorbitant computational efforts. If the system has not converged on the minimum fitness criteria before 50 generations, then the process is terminated based on the maximum generation criteria.

5.5 Conclusion

The chapter presents the detailed design methodology of the current research work in order to design Genetic-Fuzzy hybrid system. The generic architecture is implemented by utilizing the detailed design methodology as presented in this chapter. The problem is to decide suitable career field for the students based on the levels of multiple intelligences he/she possess. The chapter explains the steps for designing algorithm in order to achieve genetic rule learning in an optimized way. The interface is designed for the human expert to provide his/her own decision regarding appropriate class of the students based on the specific training data set. In order to determine relative importance of the different data elements, fuzzy decision making approach is utilized. The score of the questionnaires based on the Theory of Multiple Intelligence (MI) is collected as the pre-requisites and later the score of different intelligences of the individual students is classified into four distinct levels such as “Very High”, “High”, “Mid” and “Low”. The fuzzy membership functions define the ranges of the above stated measures of intelligence. As a result linguistic representation is achieved using the truth value. The foundation of GA system is to design the architecture of the GA rule. Here, novel strategy for architecture of rule is implemented by mapping five intelligences namely Logical, Verbal, Interpersonal, Intrapersonal and Musical/Visual along with level such as “Very High”, “High”, “Mid” and “Low”. The chapter explains significance of the training data set designed for predicting career field of the students along with examples of training set used for training the system. GA starts with random rule generation and fitness of GA rule is to be calculated to check the quality of the rule and finally to select parents for next generation. The chapter discusses calculation of fitness for the GA rule by explaining
suitable examples. In case of finding non optimal solution, sequential generations of operators are applied. Here, the different types of crossover and mutation operators are designed along with the arithmetic crossover. Finally, chapter concludes with convergence criteria in order to achieve an optimal outcome. After meeting the convergence criteria, the system generates optimal outcome in form of the suitable career field for the test data given as an input in form of value of score.