5.1 INTRODUCTION

In the previous chapter, system reliability was considered as one of the major attributes in the long-range planning of complex systems. In this chapter, more concentration is given to the system reliability modelling problems. This chapter explains

i. the suitabilities of Prolog to represent reliability networks.

ii. Monte-Carlo technique for optimal redundancy allocation.

The reliability values obtained from series and parallel networks are compared with the basic operations that can be done with fuzzy sets and fuzzy concepts are also applied to reliability network modelling.

The capabilities of Prolog are illustrated by relevant examples in various reliability network configurations. The network reliability has been estimated by using the recursive nature of Prolog. This approach is very useful to the reliability engineers who apply AI technique in reliability studies.
Monte-Carlo optimization technique is used for the redundancy allocation problem for maximizing the system reliability subject to cost constraint. The algorithm is clearly described with an example. It has been proved that computational time is quite short for this approach as compared with the existing dynamic programming approach. The proposed method looks at a random sample of feasible solutions and takes the best one. This method gives the best course of action out of more possibilities. The proposed method gives the optimal solution in most cases without much special computational effort.

Similar to a stochastic process, a fuzzy process is carried out for reliability modelling of a power generator with a derated state. Linguistic values are assigned to the state transition probabilities.

5.2 PROLOG REPRESENTATION OF RELIABILITY NETWORKS AND EVALUATION OF SYSTEM RELIABILITY

A Pascal or C program defines the procedures that the program must follow to arrive at a solution or to perform some task. In Prolog, a program is not defined as a series of steps but as a description of the relationship between objects. From these descriptions, Prolog derives solutions to questions. Prolog is centered around a small set of basic mechanisms which include pattern matching, tree-based data structuring and automatic backtracking.

Prolog is not just a descriptive language. It is also a procedural language [34]. Reliability networks can be represented as a structure by using the descriptive nature
of Prolog and the network reliability can be evaluated with high degree of accuracy by using the procedural nature of Prolog.

Prolog is well suited for problems that involves objects, in particular, structured objects and relations between them. Table 5.1 shows the simple reliability networks, their tree representation and Prolog representation [35].

5.2.1 Structuring information about a reliability network

Prolog represents and manipulates structured data objects. A database about reliability networks can be represented so that each block in the network can be described by one clause. Figure 5.1 shows a reliability network with 6 blocks. Figure 5.2 shows how the information about each block can be structured. Each block represents the various components contained in it and the way in which it is connected to the next block.

5.2.2 Data abstraction

Figure 5.2 gives the structuring information about the reliability network. Each block of information can be easily accessed. All the details of implementing such a structure in a computer should be invisible to the user of the structure. The Prolog programmer can concentrate on objects and relations between them. The point of this process is to make the use of the possible information without the programmer having to think about the details of how the information is actually represented.
### Table 5.1: Prolog Representation of Reliability Networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Tree Representation</th>
<th>Prolog Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Network Diagram 1" /></td>
<td>SERIES $R_1$ (\parallel R_2)</td>
<td>SERIES ((R_1, R_2))</td>
</tr>
<tr>
<td><img src="image2" alt="Network Diagram 2" /></td>
<td>PARALLEL $R_1$ (\parallel R_2)</td>
<td>PARALLEL ((R_1, R_2))</td>
</tr>
<tr>
<td><img src="image3" alt="Network Diagram 3" /></td>
<td>PARALLEL $R_1$ (\parallel R_2) (\parallel R_3)</td>
<td>PARALLEL ([R_1, \parallel (R_2, R_3)])</td>
</tr>
<tr>
<td><img src="image4" alt="Network Diagram 4" /></td>
<td>PARALLEL $R_1$ SERIES ((R_2, R_3)) (\parallel R_4)</td>
<td>PARALLEL ([R_1, \parallel (R_2, R_3), R_4])</td>
</tr>
</tbody>
</table>
FIGURE 5.1 RELIABILITY NETWORK

FIGURE 5.2 STRUCTURING INFORMATION ABOUT THE NETWORK
Consider the sample reliability network given in Figure 5.1. Each block is a collection of pieces of information. Each block can be treated as single data object. A relation regarding the reliability network structure can be defined through which the user can access a particular block of a network without knowing the details of Figure 5.2. The structured information about the network given in Figure 5.2 can be stored in the database by the clause:

```
network(blocka(series(R1,R2)))
    blockb1(series(blocka,parallel(R3,R4))),
    blockd(series(blockb1,parallel(R5,series(R6,R7))))).
```

The block D represents the entire network recursively. This kind of representation enables evaluation of system reliability totally or stagewise recursively.

5.2.3 Evaluating system reliability

The structure of the network (Figure 5.1) is represented in Prolog as follows:

1. The number of elements in the network (Ne) are counted

2. Each element in the structure is named as R1, R2, ...

3. Each element name with its reliability value is stored in the database ELEMENT-DETAIL

4. The block information of Figure 5.2 is represented as follows:
i. Elements in a structure are grouped in pairs and named as Block A, Block B respectively.

ii. Each block information is entered interactively.

Table 5.2 shows the sample network representation.

5. System reliability is evaluated recursively. From Table 5.2, it is clear that the total system reliability is obtained from Block identification D. Recursion is a common programming technique in Prolog. The result of one block is needed by the next and the result of that block in turn, is needed by the next.

TABLE 5.2 SAMPLE RELIABILITY NETWORK REPRESENTATION

<table>
<thead>
<tr>
<th>Block</th>
<th>Elements</th>
<th>connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>R1, R2</td>
<td>series</td>
</tr>
<tr>
<td>B</td>
<td>R3, R4</td>
<td>parallel</td>
</tr>
<tr>
<td>B1</td>
<td>BLOCK A, BLOCK B</td>
<td>series</td>
</tr>
<tr>
<td>C</td>
<td>R6, R7</td>
<td>series</td>
</tr>
<tr>
<td>C1</td>
<td>R5, BLOCK C</td>
<td>parallel</td>
</tr>
<tr>
<td>D</td>
<td>BLOCK B1, BLOCK C1</td>
<td>series</td>
</tr>
</tbody>
</table>
The above steps can be represented as a Prolog segment given in Appendix 1. The goal,

?- network_representation

will run the Prolog segment and the element and block details are added into the database interactively. The blocks are represented in the database as shown in the Table 5.2. The goal,

?- get_reliability_value('D',X)

will result in total system reliability and X will be instantiated to that value. The goal,

?- block('B1', X,Y,Z)

will yield the result as follows:

X = 's', Y = A, Z = B

The above result means that blocks A and B are connected in series and are given the block identification B1. In turn, the goal,

?- block('A', X,Y,Z)

will yield the result as follows:

X = 's', Y = R\textsuperscript{1}, Z = R\textsuperscript{2}

and so on.
In this way, we can represent the network and also we can evaluate the network reliability, using Prolog. The structure of the network can be retrieved in any way. Prolog backtracks through the database automatically and evaluates the system reliability recursively.

5.3 REDUNDANCY OPTIMIZATION [9,10,11,36]

The reliability of a system can be increased by using redundancies at various stages in the system. But there are always some inherent constraints on the system such as cost, weight and volume that put limitations on the amount of redundancy to be used for each subsystem. One usually seeks an optimum solution to this allocation problem under the specified constraints. The technique presented here solves problems such as maximizing reliability subject to cost constraint, which need not be linear.

5.3.1 Problem formulation

Systems which have to perform their intended operations must be designed to ensure maximum reliability throughout their lifetime. In addition, certain constraints like component costs and budget have to be taken into account. For the present purpose, a series system may be considered. The series system is made up of \( N \) elements, all of which must be operating for the system to be able to function. The objective is to maximize the reliability of the system by adding redundancies without exceeding the available capital cost \( C \).
The system reliability $R_S$ over a given period is the product of the reliabilities of the elements

$$ R_S = \prod_{i=1}^{N} R_i $$  \hspace{1cm} (5.1)

The problem of selecting maximum redundancy then becomes one of determining a vector

$$ n = (n_1, n_2, \ldots, n_N) $$

with $n_i$ positive integers and solving the problem

$$ \text{Maximize } R_S(n) = \prod_{i=1}^{N} R_i $$

$$ \text{Subject to } \sum_{i=1}^{N} C_{ij} n_i \leq C_j \quad j=1,2,\ldots $$

In certain problems the constraint may be defined in terms of reliability and the function to be optimized is the cost. The problem can therefore be written as

$$ \text{Minimize } C(n) = \sum_{i=1}^{N} C_i(n_i) $$

$$ \text{Subject to the constraint } R(n) \geq R_0 $$

The problem considered has the following assumptions.

i. Each element has at least one redundancy unit.

ii. The maximum number of redundant units that can be added to each element is known.
iii. The data for the reliability $R_j(K_j)$ and the cost $C_j(K_j)$ for the $j$th component ($j=1,2,3,...,N$) given $K_j$ parallel units are known.

The objective is to determine the number of parallel units, $K_j$ in component $j$ that will maximize the reliability of the system without exceeding the allocated capital.

The above problem can be solved by dynamic programming [9] and the problem formulation is

$$\max f_N(y_N) = K_N \quad \{ R_N(K_N) \}$$

$$C_N(K_N) \leq y_j$$

$$\max f_j(y_j) = K_j \quad \{ R_j(K_j) \cdot f_{j+1}(y_j - C_j(K_j)) \}$$

$$C_j(K_j) \leq y_j$$

$$j = 1,2,...,N-1$$

where $f_j(y_j)$ be the total optimal reliability of components $j, j+1, ... N$ given the capital $y_j$.

The algorithm presented below adopts the cost criterion as follows:

$$P = MC - \left( \sum_{i=1}^{N} C_j(K_j - MC) \right)^2$$

(5.5)

where $C_j(K_j)$ is the cost associated with $i$, given the number of parallel redundant units $K_i$ and $MC$ is the total capital.

5.3.2 Proposed algorithm

A method for reliability optimization through random selection of redundancy have been proposed. The plan [37] is
to create a large feasible solution region and need in hundreds of feasible random solutions, always storing the best answer so far and recentering the solution region about the best answer so far. Then the solution region is reduced in size and the process is repeated. This is done over and over again until the solution region slide around and surround the solution and then find the solution.

STEP 1:

Read in the number of elements in the system (N) and read in the maximum number redundancy units that can be added to each element (MR) and initialize the first seed value (IY).

STEP 2:

Read in the reliability values in matrix R(I,J) and read in the costs in matrix IC(I,J), I=1,2,...MR and J=1,2,...,N.

STEP 3:

Initialize BIG = 0 and initialize M = -9999 (assumed lowest value)

STEP 4:

Set the lower bound of the solution region equal to 1 and the upper bound equal to MR for all the stages (N) (i.e.) IB(I)=1 and NN(I)=MR, I=1,2,...N. Center the solution region as follows. If MR is odd then NL=MR/2+1, if MR is even then NL=MR/2 (i.e.) IA(I)=NL. The array IA stores the best answer so far.
STEP 5:

Determine hundred (IZ, arbitrarily chosen which can be refined) feasible random solutions at each solution region. Set J = 1.

STEP 5a:

Control the size of the solution region, making them smaller each time by the following way. Set I=1.

STEP 5b:

Set K=1.

STEP 5c:

If (IA(K) - NN(K)/2**J)<IB(K) set the lower bound as L(K)=IB(K) otherwise set L(K)=IA(K) - NN(K)/2**J. Determine the width of the solution region as follows:

IF(IA(K)+NN(K)/2**J)>NN(K), the width of the region IAU(K)=NN(K)-L(K), otherwise IAU(K)=IA(K)+NN(K)/2**J-L(K).

This step controls the boundaries and size of ever-narrowing solution region and makes sure that region does not go below or above the lower and upper bound of the feasible solution region. Generate one random number using the function RANDOM(IY,JY,U). U contains the uniformly distributed random number between 0 and 1. Store this value in the array IX as

\[ IX(K) = L(K)+\text{INT}(U*IAU(K)+0.49999) \]

Modify the seed value as IY=JY.
STEP 5d:

Increment K by 1
If K ≤ N go to step 5c, else go to step 6.

STEP 6:

The array IX now contains the randomly selected redundant units that can be added to each element in the system. Set IPF = MC where MC is the total cost available.

STEP 7:

\[ \text{Compute ISUM} = \sum_{K1=1}^{N} \sum_{L1=IX(K2)} IC(L1,K1), L1=IX(K2), K2=1,2,..MR \]

Evaluate IPF = IPF - (ISUM-MC)**2

If IPF > M go to STEP 8 otherwise go to STEP 9.

STEP 8:

Set M=IPF and store the elements of IX array in IA array.

IA array contains the best solution so far.

STEP 9:

Increment I by 1.

If I ≤ IZ go to STEP 5b, else go to STEP 10.
STEP 10:

Increment J by 1.
If J < NL go to STEP 5a, else go to step 11.

STEP 11:

Compute optimal reliability

\[ R_{\text{MAX}} = \prod_{I=1}^{N} R(\text{IM},I) \quad \text{IM} = \text{IA}(J), \quad J=1,2,...N \]

Print the results.

5.3.3 Discussions

The above algorithm is tested with the system containing \( n \) series elements using a PRIME 2250 system.

The time taken to obtain the optimal result using dynamic programming approach and using the proposed method are compared for 3 elements and 4 elements series system and is shown in Table 5.3.

Example 5.1

The design of an electronic device consisting of three main components may be considered [38]. The three components are arranged in series so that the failure of one component will cause the failure of the entire device. The reliability of the device can be improved by installing standby units in each component. The design calls for using one or two standby units, which means that each main component may include up to 3 units in parallel. The total capital available is ten thousand dollars. The data are summarized in Table 5.4.
### TABLE 5.3 COMPUTING TIME COMPARISONS

<table>
<thead>
<tr>
<th>Elements</th>
<th>Approach</th>
<th>Time taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Dynamic programming</td>
<td>9 secs.</td>
</tr>
<tr>
<td></td>
<td>Proposed method</td>
<td>1 sec.</td>
</tr>
<tr>
<td>4</td>
<td>Dynamic programming</td>
<td>15 secs.</td>
</tr>
<tr>
<td></td>
<td>Proposed method</td>
<td>2 sec.</td>
</tr>
</tbody>
</table>

### TABLE 5.4 SAMPLE DATA FOR REDUNDANCY OPTIMIZATION PROBLEM

<table>
<thead>
<tr>
<th>$K_j$</th>
<th>$j = 1$</th>
<th>$j = 2$</th>
<th>$j = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_1$</td>
<td>$C_1$</td>
<td>$R_2$</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
<td>1.1</td>
<td>0.7</td>
</tr>
<tr>
<td>2</td>
<td>0.8</td>
<td>2.2</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>0.9</td>
<td>3.3</td>
<td>0.9</td>
</tr>
</tbody>
</table>
The above problem is solved using the proposed algorithm. With $C$ having a value of 10 and $k_1$, $k_2$, and $k_3$ get values of 2, 1 and 3 respectively and the maximum reliability is 0.5040. This matches well with the results given in the reference [38].

5.4 FUZZY RELIABILITY MODELLING

In this section, fuzzy concepts are applied to reliability network modelling.

Reliability of elements in a network depends on numerous factors. To evaluate the probability associated with the failure in an analytic manner considering all such factors would become very difficult. As an alternative, fuzzy modelling has been applied for subjective evaluation which replaces the analytic approach. In addition, the approach as presented in this section is of another kind which has not been explored so far in power system environments.

5.4.1 Comparison between reliability networks and basic operations on fuzzy sets

In chapter 2, basic operations on fuzzy sets like union, intersection operations and algebraic sum etc., have been discussed. The results obtained from these operations can be compared with the reliability values obtained from the networks. For two elements $R_1$ and $R_2$ connected in series, the system reliability is given by $R_s = R_1 \times R_2$ and $R_s$ must be less than or equal to the minimum of $R_1$ and $R_2$. This result resembles the intersection operation on fuzzy sets. If these elements are connected in parallel, the reliability value will be $R_p = 1 - (1-R_1) (1-R_2)$, which
resembles the grade of membership obtained from the probabilistic algebraic sum of two fuzzy subsets. \( R_p \) must be greater than or equal to the maximum of \( R_1 \) and \( R_2 \). This is similar to the union operation on fuzzy sets.

If the operation has to be done with three fuzzy subsets, it is as given below:

\[
g(A,B,C) = (A \land B \land C) \lor (A \land B) \lor (A \land B \land C)
\]

where \( g(A,B,C) \) is the grade of membership obtained from the above fuzzy operation. The network for the above operation can be shown as in Figure 5.3.

![Figure 5.3 Sample Reliability Network Representing \( g(A,B,C) \)](image)

The reliability value obtained from the above network must be greater than or equal to \( g(A,B,C) \)
If we have fuzzy elements in a reliability network, that is, the reliability values of those elements are not known precisely, we can assign linguistic variables such as Low, Medium, High etc., as reliability values and then we can evaluate the total network reliability. The final result obtained is also fuzzy in nature yet it conveys a meaningful solution if we quantify the linguistic variables. The fuzzy grades of membership of linguistic variables low, medium and high are defined by the following fuzzy sets [1,39].

Low: \[ (0.5,0.2), (0.7,0.3), (1.0,0.4), (0.7,0.5), (0.5,0.6) \]

Medium: \[ (0.5,0.4), (0.7,0.5), (1.0,0.6), (0.7,0.7), (0.5,0.8) \]

High: \[ (0.5,0.7), (0.7,0.8), (0.9,0.9), (1.0,1.0) \]

Therefore, the fuzzy sets for very low, very medium, and very high are as follows [39]:

Very Low = Low^2

Very Low: \[ (0.25,0.2), (0.49,0.3), (1.0,0.4), (0.49,0.5), (0.25,0.6) \]

Very Medium: \[ (0.25,0.4), (0.49,0.5), (1.0,0.6), (0.49,0.7), (0.25,0.8) \]

Very High: \[ (0.25,0.7), (0.49,0.8), (0.81,0.9), (1.0,1.0) \]
Similar to a stochastic process, a fuzzy process can be defined as follows [23]:

A finite process in discrete time with a discrete state space $Q = \{q_1, q_2, \ldots, q_n\}$ is called a finite fuzzy process if it satisfies the following conditions.

i) The matrix $F$ which describes the state transition has the following form

$$
F = \begin{bmatrix}
q_1 & q_2 & q_r \\
q_1 & f_{11} & f_{12} & \cdots & f_{1r} \\
q_2 & f_{21} & f_{22} & \cdots & f_{2r} \\
& \cdots & \cdots & \cdots & \cdots \\
q_r & f_{r1} & f_{r2} & \cdots & f_{rr}
\end{bmatrix}
$$

where $0 < f_{ij} \leq 1$ denotes the grade of membership of state transition from state $q_i$ to state $q_j$. This matrix will be called the fuzzy state transition matrix of the fuzzy process.

ii) Let $A$ be a fuzzy set defined on $S$, and

$$
\begin{align*}
\omega_A^{(0)} &= [ \varphi_A^{(0)} q_1 \varphi_A^{(0)} q_2 \cdots \varphi_A^{(0)} q_r ] \\
\omega_A^{(n)} &= \omega_A^{(0)} \cdot p^n = \omega_A^{(0)} [ f_{ij}^{(n)} ]
\end{align*}
$$
where

\[ f(n) = \max_{i,j} \min_{\text{over all parallel paths series paths}} \left( \mu_{ij} \right) \]

\[ \mu_{ij} \text{ = the set of grades of membership of state transitions of a path from } q_i \text{ to } q_j. \]

An example is shown in the next section for the application of a fuzzy process in the reliability modelling of power generation systems.

5.4.2 Markov modelling of a power generator with a derated State - Linguistic approach

In this model a power generator is said to have three states [40]:

i) fully operational (full power generation)

ii) partial operation (power generation at a derated level)

iii) failed (no power output)

For example, at a coal-fired power station, the generator derated state may occur due to failure of some of the unit pulverizers.

The state space diagram for the above conditions is given in Figure 5.4. The generating system can either be derated or failed completely and can be repaired. Here it
Figure 5.4 State Space Diagram
is assumed the state transition probability are known imprecisely and defined in terms of linguistic variables.

The fuzzy state transition matrix is given below.

\[
F = \begin{bmatrix}
H & M & VL \\
M & VH & M \\
L & M & VL 
\end{bmatrix}
\]

The tree diagram for this state transition matrix is shown in Figure 5.5. This tree diagram explains the status of the power generator after two steps, given that the fuzzy process started in state 1, state 2 and state 3 respectively.

That is,

\[
F^2 = \begin{bmatrix}
H & M & M \\
M & VH & M \\
M & M & M 
\end{bmatrix}
\]

If the initial state designator \( w_A^{(0)} \) is defined as,

\[
w_A^{(0)} = [ M \ H \ L ]
\]

then the state designator at \( t = 2 \) can be obtained as follows:

\[
w_A^{(2)} = w_A^{(0)} F^2
\]
FIGURE 5-5 TREE DIAGRAM
By using the fuzzy basic operations,

\[ w_B^{(2)} = [ M \ H \ M ] \]

The first element in the above vector is obtained as follows:

\[
(M \cap VH) \cup (H \cap M) \cup (L \cap H) = [M]
\]

Using the quantitative measures of linguistic variables given in section 5.4.1 the following \( w_B^{(2)} \) vector can be obtained.

\[
\begin{align*}
\hat{w}_A^{(2)} &= \{ \begin{array}{ll}
(0.5,0.4), & (0.7,0.5), & (1.0,0.6), & (0.7,0.7), & (0.5,0.8) \\
(0.5,0.7), & (0.7,0.8), & (0.9,0.9), & (1.0,1.0) \\
(0.5,0.4), & (0.7,0.5), & (1.0,0.6), & (0.7,0.7), & (0.5,0.8)
\end{array} \}
\end{align*}
\]

\[
\hat{w}_A^{(2)} \approx [0.6 \ 1.0 \ 0.6]
\]

The approximate \( \hat{w}_A^{(2)} \) vector contains the values those have highest grade of membership in the quantitative definitions of linguistic terms and expresses that there is a high possibility of the system being in state 2 (partial operation state) after two transitions.
The fuzzy steady-state designation vector is determined by the following equation:

\[ u_i \setminus f_{ij} = \pi_i \quad i, j = 1, 2, \ldots, k \]

where \( k \) is the total number of states.

that is, for the above considered 3-state problem,

\[
\begin{align*}
(\pi_1 \setminus f_{11}) \lor (\pi_2 \setminus f_{21}) \lor (\pi_3 \setminus f_{31}) &= \pi_1 \quad (1) \\
(\pi_1 \setminus f_{12}) \lor (\pi_2 \setminus f_{22}) \lor (\pi_3 \setminus f_{32}) &= \pi_2 \quad (2) \\
(\pi_1 \setminus f_{13}) \lor (\pi_2 \setminus f_{23}) \lor (\pi_3 \setminus f_{33}) &= \pi_3 \quad (3)
\end{align*}
\]

Since the operations on the left hand side (LHS) of the above equations are fuzzy operations, to solve them, the following procedure is used.

**STEP 1:**

Start with an initial approximation for \( \pi_1, \pi_2, \pi_3 \).

**STEP 2:**

Compute LHS of equations (1), (2) and (3). Let \( \epsilon \) be the tolerance limit.

If \( (\text{LHS}_i - \text{RHS}_i) \leq \epsilon \) for \( i = 1, 2, 3 \)

then the obtained \( \pi_i \) vector is the fuzzy steady-state designation vector. Stop.

else perturb the values of \( \pi_1, \pi_2, \) and \( \pi_3 \).

Go to Step 2.
For example if $\pi_1$ vector is $[0.6 \ 1.0 \ 0.4]$, then the fuzzy steady-state designation vector is given as,

$$[0.6 \ 1.0 \ 0.6]$$

Linguistic approach is an useful aid in decision making, modelling and planning relating to real world complex problems. It is of great opinion that even in a fuzzy environment, with the knowledge of experts it is possible to attach relevance to the long-range behaviour of dynamical systems.

5.5 CONCLUSION

Reliability network is viewed in a different way in this chapter. This chapter emphasizes the fact that Prolog is suitable for computing the reliability value recursively. Prolog acts as a descriptive language while representing a reliability network and it acts like a procedural language, while evaluating the reliability value. This approach is very useful to reliability engineers who apply AI technique in reliability studies.

A simulation technique has been developed for dynamic redundancy allocation problem. The proposed method looks at a random sample of feasible solutions and takes the best one. This method gives the best course of action out of more possibilities. The proposed method gives the optimal
solution in most cases without much computational effort. The special features of this method are,

- decision variables are treated as integers. Therefore no rounding off at the end is needed.

- method is easy

- this method provides the optimal solution in many cases and a near optimum solution in others.

- this method is applicable to any system, as long as the analytic expression for the reliability $R_s$ is available.

The basic operations on fuzzy sets are compared with the reliability values obtained from the networks. Fuzzy Markov modelling of a power generator with a derated state has been developed. In the next chapter, energy availability modelling of power systems is carried out and an algorithm has been explained to estimate uncertainty importance factors of generating units.