4.1 GENETIC ALGORITHMS

The essential ideas were created by Holland, while the practically of utilizing the GA to solve complex issues was illustrated. Genetic Algorithms (GAs) is a soft computing approach.

GA is a maximization procedure. The issue it addresses normally has a huge search space with likely various nearby maxima inside it. The GA procedure needs to guarantee that it is not caught at neighborhood maxima, so that toward the end of the procedure it might locate the worldwide maxima regardless of the possibility that the worldwide maximum is not returned, we may expect a nearby approximation of it as the result of the GA procedure.

To accomplish this, GA works on a set of an arrangement of answers for the given issue occurrence, and develops it through number of generations. The optimization procedure stops when some predefined end condition is fulfilled. At each intermediate stage, the old generation is supplanted by the new generation. The people of the number of inhabitants in an era are prepared with the assistance of various GA administrators in such a way that the nature of the new era, when all is said in done is enhanced in examination with old generation. Thus, we acquire better and better arrangement will be returned by the GA procedure.
4.1.1 Pseudo code for Genetic Algorithm
Following is the pseudo code for the basic procedure of Genetic Algorithm:
BEGIN
- INITIALIZE POPULATION with random candidate solution.
- EVALUATE (by encoding process) each candidate fitness value
- REPEAT UNTIL termination condition is satisfied
begin
• SELECT parents
• RECOMBINE pair of parents
• MUTATE the resulting offspring
• SELECT INDIVIDUAL or the next generation
end
END

4.1.2 Generation Cycle or Genetic Algorithm Cycle
The Genetic Algorithm Cycle is defined as follows:
1. Start: “Generate random population of n chromosomes”
2. Fitness: “Evaluate the fitness function $F(x)$ of each chromosome $x$ in population”.
3. New Population: “Create a new population by repeating following steps until the new population is computed”: 

---

Figure 4.1: Basic Procedure for Genetic Algorithm
(a). **Selection**: “Select two parent chromosomes from population according to their fitness” i.e. better the fitness, better the chance to be selected.

(b). **Crossover**: “Crossover the parents to form new offspring (children)”. If no Mutation was performed then the offspring is the exact copy of parents.

(c). **Mutation**: Mutation with a mutation probability, mutate new offspring at each rows.

(The mutation is the changing of structure in a gene, resulting in variant form than its parents parts).

(d). **Accepting**: “Place new offspring in the new population”.

4. **Replace**: Use new generated population for choose further population.

5. **Test**: “If the termination condition is satisfied, stop and return the best solution in current population”.

6. **Loop**: Go to step 2.

---

**Figure 4.2: Generation Cycle**
4.1.3 Flow chart of Genetic Algorithm

Figure 4.3: Genetic Algorithm Flow Chart
4.2 PARAMETERS AND ASSUMPTION TAKEN FOR THE ALGORITHM

The following parameters are taken for the implementation of Genetic algorithm:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_NFITNESS</td>
<td>Cumulative Fitness Value</td>
</tr>
<tr>
<td>CFLAG</td>
<td>The clamping flag</td>
</tr>
<tr>
<td>CHILD1</td>
<td>2-D matrix of first kid</td>
</tr>
<tr>
<td>CHILD2</td>
<td>2-D matrix of second kid</td>
</tr>
<tr>
<td>FFREQ</td>
<td>The flag frequency</td>
</tr>
<tr>
<td>FGGENS</td>
<td>The flag gens matrix initially set to -1.</td>
</tr>
<tr>
<td>FITNESS</td>
<td>The fitness value vector</td>
</tr>
<tr>
<td>FPERIOD</td>
<td>Initialize flag period</td>
</tr>
<tr>
<td>GENERATIONS</td>
<td>The maximum number of generation allowed to run</td>
</tr>
<tr>
<td>MARKER</td>
<td>To determine the indices of roulette wheel for the selection of parents.</td>
</tr>
<tr>
<td>NFFREQ</td>
<td>The Un flag frequency</td>
</tr>
<tr>
<td>P_INDICES</td>
<td>To find the indices of the parent</td>
</tr>
<tr>
<td>PARENT1</td>
<td>2-D matrix of first parent</td>
</tr>
<tr>
<td>PARENT2</td>
<td>2-D matrix of second parent</td>
</tr>
<tr>
<td>PROB_RECOMB</td>
<td>The probability of crossing over.</td>
</tr>
<tr>
<td>PSIZE</td>
<td>The size of the population</td>
</tr>
<tr>
<td>RCOST</td>
<td>Total relevant inventory cost</td>
</tr>
<tr>
<td>RECOMBINATION</td>
<td>The Crossover type</td>
</tr>
<tr>
<td>REPOSITORIESFLAG</td>
<td>If 1, then draw uniform crossover and mutation mask.</td>
</tr>
<tr>
<td>ROULETTE</td>
<td>Stochastic Universal Sampling i.e. Roulette Wheel Sampling. Select n parents by spinning a Roulette wheel within equally spaced pointer just one.</td>
</tr>
</tbody>
</table>
SIG : Standard deviation of fitness value
SSCALINGCOEFF : The sigma scaling coefficient i.e. 1
SSCALINGFLAG : Sigma Scaling flag to adjust the fitness of each individual, and used this adjusted fitness when selecting the parent of that generation.
VFLAG : The visualization of bit frequency, if zero then don’t visualize bit frequency
VRFLAG : If 1 then display the details of each generation.

It is very difficult to show each steps of algorithm with actual values of population, generation with other related parameters. So for this here we are defining the algorithm with minimum population size, minimum iterations/generations and other parameter values as follows:

PSIZE = 4; // Size of the Population.
GENERATIONS = 2; // Maximum no of generations allow to run.
RCOST = 5; // Total Relevant inventory cost.
FITNESS [PSIZE]; // Length of the Fitness value vector.
C_NFITNESS [PSIZE]; // Cumulative Fitness value vector.
POP [PSIZE] [RCOST]; // Population Matrix.
ROULETTE = 1; // Initial value of Roulette wheel variable.
4.3 PROCEDURE USED FOR GENETIC ALGORITHM

4.3.1 Representation of Chromosomes
The representation of chromosomes in GAs has profound effect on the execution of GA-based function. There are diverse strategies for representation of chromosomes like paired encoding, value encoding, permutation encoding, tree encoding and so on. The most generally utilized encoding is binary encoding proposed by Holland. In this strategy, the estimation of individual is encoded as bit string comprises of binary values either 0 or 1.

The population is a PSIZE by RCOST matrix of randomly generated Boolean values

\[
\text{POP} \leftarrow \text{Random (PSIZE, RCOST)} < .5
\]

So the population matrix will be calculated as

<table>
<thead>
<tr>
<th>POP</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 0</td>
<td>(\rightarrow) 4</td>
</tr>
<tr>
<td>0 1 0 1 1</td>
<td>(\rightarrow) 3</td>
</tr>
<tr>
<td>0 1 0 1 1</td>
<td>(\rightarrow) 3</td>
</tr>
<tr>
<td>0 0 1 1 0</td>
<td>(\rightarrow) 2</td>
</tr>
</tbody>
</table>

4.3.2 Fitness
The fitness function is the bone marrow of each Optimization problem. In the event that fitness function is bad, then no optimization strategies help in finding the solution. Accordingly, choice of fitness function is pivotal issue. The fitness function gives a route to the GA to analyze the performance of every chromosome in the population. The GA surveys answers for their quality as per the data created by this unit and not by utilizing direct data about their structure. Given a specific chromosome, the fitness function gives back a single value, which speaks to the merit of the relating answer for the issue. Fitness assessment capacities may be complex or straightforward relying upon the optimization issue close by.
Following steps are involved for evaluate the fitness:

The initial fitness of the population will be evaluated by the initial population matrix.

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

\[
\text{K} \leftarrow 0;
\text{for } I \leftarrow 1 \text{ to } \text{PSIZE} \text{ do}
\hspace{1em} \text{for } J \leftarrow 1 \text{ to } \text{RCOST} \text{ do}
\hspace{2em} \text{K} \leftarrow \text{K} + \text{POP}[I][J];
\hspace{1em} \text{end for}
\hspace{1em} \text{FITNESS}[I] \leftarrow \text{K};
\text{end for}
\]

The vector of fitness values returned must be of dimensions 1 X PSIZE i.e.

\[
\text{FITNESS} = 4 \hspace{0.5em} 3 \hspace{0.5em} 3 \hspace{0.5em} 2
\]

// To Calculate the cumulative fitness.
//Firstly evaluates the standard deviation STDEV of fitness value FITNESS.
Initialize COUNTFITNESS \leftarrow \text{PSIZE} - 1;
SIG = \text{STDEV (FITNESS [ ])};
STDEV(real FITNESS[ ])
{
    real SUMFV \leftarrow 0.0;
    real DEV \leftarrow 0.0;
    real MEANFV;
    for I \leftarrow 1 \text{ to } \text{COUNTFITNESS} \text{ do}
        SUMFV \leftarrow SUMFV + \text{FITNESS}[I];
    end for
MEANFV ← SUMFV / COUNTFITNESS;
for I ← 1 to COUNTFITNESS do
    DEV ← DEV + POW (FITNESS [I] – MEANFV, 2);
end for
return SQRT (DEV / COUNTFITNESS);

The value of the STDEV will be returned in SIG and the value of SIG will be

\[
\text{SIG} = 0.8165
\]

Now again calculate the fitness value by analyzing the SIG (the standard deviation calculated as above).

\[
\text{if SIG} \neq 0 \text{ then}
\]

\[
\text{for I } \leftarrow 1 \text{ to COUNTFITNESS do}
\]

\[
\text{FITNESS [I] } \leftarrow 1 + (\text{FITNESS [I]} - \text{MEANFV}) / \text{SIG};
\]

end for

\[
\text{for I } \leftarrow 1 \text{ to COUNTFITNESS do}
\]

\[
\text{if FITNESS [I] } \leq 0 \text{ then}
\]

\[
\text{FITNESS [I] } \leftarrow 0;
\]

end if

end for

\[
\text{else}
\]

\[
\text{for I } \leftarrow 1 \text{ to COUNTFITNESS do}
\]

\[
\text{FITNESS [I] } \leftarrow 1;
\]

end if

Here the new generated fitness value will be generated as:

\[
\text{FITNESS} = \begin{bmatrix} 2.2247 & 1.0000 & 1.0000 & -0.2247 \end{bmatrix}
\]

\[
\text{if it is less than zero then make it zero.}
\]

\[
\text{FITNESS} = \begin{bmatrix} 2.2247 & 1.0000 & 1.0000 & 0 \end{bmatrix}
\]
// Calculate the fitness probability F_PROBABILITY[ ] and cumulative fitness value C_NFITNESS[ ]

for \( I \leftarrow 1 \) to COUNTFITNESS do
  F_PROBABILITY[I] \leftarrow FITNESS[I] / SUMFV;
end for

F_PROBABILITY = 0.5266  0.2367  0.2367  0

C_NFITNESS[1] = F_PROBABILITY[1];

for \( I \leftarrow 1 \) to COUNTFITNESS – 1 do
  C_NFITNESS[I+1] = F_PROBABILITY[I+1] + C_NFITNESS[I];
end for

Normalize the fitness values and then create an array with the cumulative normalize fitness value:

| C_NFITNESS | 0.5266 | 0.7633 | 1.0000 | 1.0000 |

The distribution and calculation of the fitness values and normalize fitness values are shown in following table:

**Table 4.1: Normalize Fitness Value**

<table>
<thead>
<tr>
<th>Population</th>
<th>FITNESS</th>
<th>SIG (Standard Deviation)</th>
<th>New Generated Fitness</th>
<th>F_PROBABILITY</th>
<th>C_NFITNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>11110</td>
<td>4</td>
<td></td>
<td>2.2247</td>
<td>0.5266</td>
<td>0.5266</td>
</tr>
<tr>
<td>01011</td>
<td>3</td>
<td>0.8165</td>
<td>1.0000</td>
<td>0.2367</td>
<td>0.7633</td>
</tr>
<tr>
<td>01011</td>
<td>3</td>
<td></td>
<td>1.0000</td>
<td>0.2367</td>
<td>1.0000</td>
</tr>
<tr>
<td>00110</td>
<td>2</td>
<td></td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
4.3.3 Selection

The selection operator chooses chromosomes from the present era to be parents for the next generation. The issue is the means by which to choose the chromosomes/individuals. As indicated by Darwin's hypothesis of evolution the best ones survive to create a new offspring. Parents are chosen in pairs. The different sorts of choice strategies which are use to choose the chromosomes are Uniform Selection, Roulette Wheel Selection, Tournament Selection etc.

4.3.3.1 Roulette Wheel Selection

Roulette Wheel is one of the traditional selection techniques. The commonly used reproduction operator is the proportionate reproductive operator where a string is selected from the mating pool with a probability proportional to the fitness. “The principle of Roulette selection is a linear search through a Roulette wheel with the slots in the wheel weighted in proportion to the individual’s fitness values. A target value is set, which is a random proportion of the sum of the fitnesses in the population. The population is stepped through until the target value reached. This is only a moderately strong selection technique, since fit individuals are not guaranteed to be selected for, but somewhat have a greater chance” [80]. A fit individual will contribute more to the target value, but if it does not exceed it, the next chromosome in line has a chance, and it may be weak. It is essential that the population not be sorted by fitness, since this would dramatically bias the selection.

The Roulette process can also be explained as follows: “The expected value of an individual’s fitness divided by the actual fitness of the population. Each individual is assigned a slice of the Roulette wheel, the size of the slice being proportional to the individual’s fitness”. The wheel is spun N times, where N is the number of individuals in the population. “On each spin, the individual under the wheel’s marker is selected to be in the pool of parents for the next generation”[80].
This method is implemented as follows:

1. “Sum the total expected value of the individuals in the population”. Let it be T.

2. Repeat N times:
   (i) Choose a random integer “r” between 0 and T.
   (ii) “Loop through the individuals in the population”, sum the expected values, until the sum is greater than or equal to “r”. The individual whose expected value puts over this limit is the one selected.

Following steps are involved for implement Roulette wheel selection. For Roulette wheel selection following parameters will be used:

\[
\text{MARKER} = \begin{bmatrix} 0.5018 & 0.7518 & 1.0018 & 1.2518 \end{bmatrix}
\]

If the value of MARKER is greater than one then subtract one i.e. MARKER - 1 from each value which is greater than one.

So in this case then new MARKER matrix will be

\[
\text{MARKER} = \begin{bmatrix} 0.5018 & 0.7518 & 0.0018 & 0.2518 \end{bmatrix}
\]
4.3.4 Crossover

After reproduction simple crossover may continue in two stages. In the first place, members of the recently imitated strings in the mating pool are mated at random. Second every combine of chromosomes undergoes crossing over.

To determine first and second parent we calculate the parent indices. For calculating the parent indices, we are using the method of histogram count i.e. histc(x, bin_range). The function histc ( , ) counts the number of values in x that are within each specified bin_range.

Here, we find parent indices by histc (MARKER, [1 C_NFITNESS]).

\[ \text{[TMP P_INDICES]} = \text{histc(MARKER, [1 C_NFITNESS])} \]

From Table No 4.1 and Initial Population we will find the count in the matting pool TEMP, which is shown in following table:

<table>
<thead>
<tr>
<th>Population</th>
<th>C_NFITNESS</th>
<th>MARKER</th>
<th>String No.</th>
<th>TEMP Count in the Matting Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>11110</td>
<td>0.5266</td>
<td>0.5018</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>01011</td>
<td>0.7633</td>
<td>0.7518</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>01011</td>
<td>1.0000</td>
<td>0.0018</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>00110</td>
<td>1.0000</td>
<td>0.2518</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \text{P_INDICES = P_INDICES (Random_permutation(PSIZE))} \]

\[
\begin{align*}
\text{TMP} &= 3 \quad 1 \quad 0 \quad 0 \\
\text{P_INDICES} &= 1 \quad 2 \quad 1 \quad 1 \\
\text{P_INDICES} &= 1 \quad 1 \quad 2 \quad 1
\end{align*}
\]
By this method the first parent PARENT1 and second parent PARENT2 will be
determine by each mating pair as:

Determine the first parents of each mating pair

We will calculate first parent i.e. PARENT1 from following code,

```python
for I \leftarrow 1 \text{ to } \text{PSIZE} / 2 \ do
    temp \leftarrow \text{P\_INDICES}[I];
    for J \leftarrow 1 \text{ to } \text{RCOST} \ do
        \text{PARENT1}[I][J] \leftarrow \text{POP}[\text{temp}][J];
    end for
end for

for I \leftarrow 1 \text{ to } \text{PSIZE} / 2 + 1 \text{ to } \text{PSIZE} \ do
    K \leftarrow 1;
    for J \leftarrow 1 \text{ to } \text{RCOST} \ do
        \text{PARENT1}[I][J] \leftarrow \text{PARENT1}[K][J];
    end for
    K \leftarrow K + 1;
end for
```

To determine the first parent we find the parent indices that is for the first parent we
consider the POP matrix rows value according to the value at first to PSIZE/2
position of the parent indices. Here the value at first position in the parent indices is 1
and the value at second position in the parent indices is also 1 (because here PSIZE/2
will be 2 in this example). So in the first parent i.e. PARENT1 matrix we take the
value of first row of POP matrix for the both rows of PARENT1 matrix.

\[
P\_\text{INDICES} = \begin{bmatrix} 1 & 1 & 2 & 1 \end{bmatrix}
\]

\[
\text{POP} =
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\]
PAREN1 = POP (P_INDICES (1:PSIZE/2)).

\[
\text{PAREN1} = \\
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

Determine the second parents of each mating pair:

We will calculate second parent i.e. PARENT2 from following code,

\[
\text{for } I \leftarrow \text{PSIZE} / 2 + 1 \text{ to } \text{PSIZE} \text{ do} \\
\quad \text{temp} \leftarrow \text{P_INDICES} [I]; \\
\quad K \leftarrow 1; \\
\quad \text{for } J \leftarrow 1 \text{ to } \text{RCOST} \text{ do} \\
\quad \quad \text{PARENT2} [K][J] \leftarrow \text{POP} [\text{temp}[J]; \\
\quad \text{end for} \\
\quad K \leftarrow K + 1; \\
\text{end for}
\]

\[
\text{for } I \leftarrow 1 \text{ to } \text{PSIZE} / 2 + 1 \text{ to } \text{PSIZE} \text{ do} \\
\quad K \leftarrow 1; \\
\quad \text{for } J \leftarrow 1 \text{ to } \text{RCOST} \text{ do} \\
\quad \quad \text{PARENT1} [I] [J] \leftarrow \text{PARENT2} [K][J]; \\
\quad \text{end for} \\
\quad K \leftarrow K + 1; \\
\text{end for}
\]

To determine the second parent we find the parent indices that is for the second parent we consider the POP matrix rows value according to the value at PSIZE/2+1 to PSIZE position of the parent indices. Here, the value at PSIZE/2+1 position i.e. at 3rd position in the parent indices is 2 and the value at PSIZE position i.e. at 4th position in the parent indices is also 1. So in the second parent i.e. PARENT2 matrix we take the value of second row of POP matrix for the first rows of PARENT2.
matrix and we take the value of first row of POP matrix for the second row of PARENT2 matrix.

<table>
<thead>
<tr>
<th>P_INDICES</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>POP</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PARENT2</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Fundamentally there are three types of crossover – Single Point Crossover, Two Point Crossover and Uniform Crossover.

4.3.4.1 Uniform Crossover

Uniform crossover is quite different. “Each gene in the offspring is created by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosomes” [25].

Here in this case the randomly generated crossover mask will be

<table>
<thead>
<tr>
<th>MASKS</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
“When there is a 1 in the crossover mask, the gene is copied from the second parent, and where there is a 0 in the mask the gene is copied from the first parent”.

A new crossover mask is randomly generated for each pair of parents. Offspring, therefore, contain a mixture of genes from each parent. The number of effective crossing point is not fixed, but will average L/2 (where L is the chromosome length).

**Implementation of Crossover:**

The Matrix for PARENT1 and PARENT2 defined as:

PARENT1 [PSIZE] [RCOST];
PARENT2 [PSIZE] [RCOST];

The child will be created by following code:

```plaintext
define for I <- 1 to PSIZE do
    for J <- 1 to RCOST do
        if MASK[I][J] = 1 then
            CHILD[I][J] <- PARENT2[I][J];
        else
            CHILD[I][J] <- PARENT1[I][J];
        end if
    end for
end for
```

Table 4.3: Generation of child matrix

<table>
<thead>
<tr>
<th>PARENT 1</th>
<th>MASK</th>
<th>PARENT 2</th>
<th>CHILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>11110</td>
<td>11111</td>
<td>01011</td>
<td>01011</td>
</tr>
<tr>
<td>11110</td>
<td>10000</td>
<td>11110</td>
<td>11110</td>
</tr>
<tr>
<td>11110</td>
<td>00000</td>
<td>01011</td>
<td>11110</td>
</tr>
<tr>
<td>11110</td>
<td>01111</td>
<td>11110</td>
<td>11110</td>
</tr>
</tbody>
</table>
To generate new population matrix

\[
\text{POP} = \text{CHILD}
\]

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

### 4.3.5 Mutation

“*Mutation prevents the algorithm to be trapped in a local minimum*”. Mutation plays the role of recovering the lost genetic materials as well as for randomly distributing genetic information. It is an insurance policy against the irreversible loss of genetic material. “*Mutation has been traditionally considered as a simple search operator. If crossover is supposed to exploit the current solution to find better ones, mutation is supposed to help for the exploration of the whole search space*”[25].

“*Mutation is viewed as a background operator to maintain genetic diversity in the population. It introduces new genetic structures in the population by randomly modifying some of its building blocks. Mutation helps escape from local minima’s trap and maintain diversity in the population. It also keeps the gene pool well stocked, thus ensuring periodicity. A search is said to be periodic if there is a non-zero probability of generating any solution from any population state*”[25].

There are many different forms of mutation for the different kinds of representation. **For binary representation**, “*a simple mutation can consist in inverting the value of each gene with a small probability*”. The probability is usually taken about 1/L, where L is the length of the chromosome. It is also possible to implement kind of hill climbing mutation operators that do mutation only if it improves the quality of the solution. Such an operator can accelerate the search; however, care should be taken, because it might also reduce the diversity in the population and make the algorithm converge toward some local optima. “*Mutation of a bit involves flipping a bit, changing 0 to 1 and vice-versa*”.
Here the mutation mask after considering probability mutation cost will be

\[
\text{MASKS} = \\
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

### 4.3.5.1 Flipping

“Flipping of a bit involves changing 0 to 1 based on mutation chromosome generated”. Firstly, a parent is considered and a mutation chromosome is randomly generated. “For a one bit (1 bit) in mutation chromosome, the corresponding bit in parent chromosome is flipped (0 to 1 and 1 to 0) and child chromosome is produced”.

For generate new population we apply the method of flipping with the help of XOR function.

The steps involved in it are as follows:

```plaintext
for I ← 0 to PSIZE do
    for J ← 0 to RCOST do
        if (POP[I][J] = 1 AND MASK[I][J] =1) OR (POP[I][J] = 0 AND MASK[I][J] =0) then
            POP[I][J] = 0;
        else
            POP[I][J] = 1;
        end if
    end for
end for
```
So after above steps the new generated population for next generation/iteration will be

\[
\text{POP} = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & \\
1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 
\end{bmatrix}, \quad \text{MASKS} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 
\end{bmatrix}
\]

After mutation and flipping a new population is generated. Apply the same procedure of GA by considering the new population as initial population for the next generation and so on.

This procedure repeats till we reach the termination condition. Here, the termination condition will be the maximum number of generation/iteration to reach the desired solution i.e. to reach the optimum cost. The above GA procedure will be repeated for both total inventory cost i.e. Case -1 and Case -2 as calculated in Inventory model.

### 4.4 ARTIFICIAL BEE COLONY

#### 4.4.1 Flow chart of the ABC algorithm

Figure 4.4 demonstrates the stream outline of the ABC algorithm. At the initial step, position of sustenance (food) source is randomly produced and the estimations of control parameters of the algorithm are doled out. The nectar measure of an underlying sustenance source counted. This step implies that the quality of beginning arrangements is computed. After retrieving the quality estimation of the underlying arrangement, employee honey bees move to her sustenance source zone to deciding another nourishment (food) source.
“Employee honey bees return into the hive and share the data about the nectar measure of their source with onlooker honey bee after finished their searching. Onlooker bees look out for the dance territory and getting the data from the employee bees”[85].

“After getting the information from employee bee, all onlooker bees memorize the position of the best nourishment source”. The following stage is to decide a neighbor nourishment source position for the onlooker bees. At that point, the nectar sum for another nourishment source has been computed once more [85].

“The progression for onlooker bees was repeated until the point when all the onlooker bees were disseminated for all food sources. The new nectar amount is assessed and compare with the pervious nectar amount. On the off chance that the new nectar amount is higher than past nectar, the new nectar amounts were kept”[85].

“The next stage is to locate the surrendered food source and deliver new position for the exhausted food source”. The entire nectar amount in the entire procedure looked at before the end procedure [85].

“If the termination criteria for this procedure are not fulfilled, this procedure repeated from the third step i.e. to decide the new nourishment positions for the employee bees until the point when the end criteria are fulfilled”. At last, the ABC algorithm ended and last food (nourishment) positions accomplished [85].
Figure 4.4: Flow chart of the ABC algorithm
4.5 ARTIFICIAL BEE COLONY ALGORITHM

ABC is an optimization algorithm based on the intelligent foraging behavior of honey bee swarm. “It is additionally called swarm based meta-heuristic algorithm”. It comprises of a queen/ruler, a couple of automatons and a large number of workers. “For food foraging workers assumes a vital part. Surrender of a source and recruitment of a nectar source are two principle methods of behavior for foraging ”[44].

ABC algorithm is separated into two parts, one part of the colony consists of employee bee and the second part is onlooker bees. “There is just a single employee bee for every food source and this representative employee bee can turns into a scout bee when the food source is consumed by the employee bee and onlooker bee”[44].

![Figure 4.5: Basic elements of foraging behavior](image)

ABC comprises of three gatherings of bees “employed artificial bees, onlooker bees, and scouts bees”. Employed bees related with particular food sources, inside the hive the dance of employed bees is viewed by onlooker bees to pick a food source, scout honey bees arbitrarily looking for food sources. In ABC, answer for the issue is given by the position of a food source and the quality (fitness) of the associated solution is
spoken to by the nectar measure of a food source. “The quantity of food sources (arrangements) is equivalent to the quantity of employed bees is equivalent to since each employed bee is related with a food source”.

4.6 COMPONENTS OF ABC MODEL

ABC model contains four main components-

1. **Initialization:** The initial food sources are randomly produced by the following equation:

\[ x_m = l_i + \text{rand} \ (0,1) \times (u_i - l_i) \] 

............... (4.1)

Where “\( u_i \) and \( l_i \) are the upper & lower bound of the solution space of objective function, \( \text{rand} \ (0,1) \) is a random number within the range \([0, 1]\)”.

Here we are considering upper bound 10 and lower bound – 10.

2. **Employed foragers:** “At a particular food source an employed forager is employed for which she is as of now misusing. About this particular source she conveys information and offers it with different bees waiting or presented in the hive with all information about the direction, the separation and the profitability of the food source”. The employed bee is assigned to one of the sources. “The quantity of the food source is equivalent to the quantity of employed bees. The bees have computed another arrangement by methods for flying to another close-by food source and hold the best solution when achieving the source”[44].

The nearby or neighbour food source \( v_{mi} \) is determined and calculated by the following equation:

\[ v_{mi} = x_{mi} + \varphi_{mi}(x_{mi} - x_{ki}) \] 

............... (4.2)

Where \( i \) is a randomly selected parameter index, \( x_k \) is a randomly selected food source, \( \varphi_{mi} \) is a random number within the range \([-1, 1]\).
The fitness is calculated by the equation 4.4, after that a greedy selection is applied between $x_m$ and $v_m$.

$$fit_m(x_m) = \frac{1}{1 + f_m(x_m)}, f_m(x_m) > 0 \quad \ldots \ldots (4.3)$$

and

$$fit_m(x_m) = 1 + |f_m(x_m)|, f_m(x_m) < 0 \quad \ldots \ldots (4.4)$$

Where, $f_m(x_m)$ is the objective function for the value of $x_m$.

3. **Food Sources/Onlooker bees:** A forager bee evaluates several properties related with the food source, for example, its closeness to the hive, taste of its nectar, amount of the energy to choose a food source. For the easiness, only a single amount represents to the quality of a food source as it relies upon different parameters specified previously. “Onlooker bees wait in the hive and settle on the choice to pick the food source in view of information from employed bees. Number of onlooker bees is the same as the employed bees and they distributed the food source based on their probability”.

The quantity of a food source is evaluated by its profitability and the profitability of all food sources. $P_m$ is determined by the formula

$$P_m = \frac{fit_m(x_m)}{\sum_{m=1}^{M} fit_m(x_m)} \quad \ldots \ldots (4.5)$$

Where, $fit_m(x_m)$ is the fitness of $x_m$. Onlooker bees search the neighborhoods of food source according to the following equation:

$$v_{mi} = x_{mi} + \varphi_{mi}(x_{mi} - x_{ki}) \quad \ldots \ldots (4.6)$$

4. **Unemployed foragers/ Scout bees:** A forager bee likewise called unemployed that searches for a food source to exploit it. It can be either an onlooker who tries to discover a food source by the assistance of employed bee's information or scout who haphazardly searches the environment.

“Scout bees are responsible for finding new food source, the new nectar and carrying out random searches”. The scout bees were figured a new solution arbitrarily when the source does not enhance after certain number of cycles.
“Source that scout bee found has been replaced the current solution if the profitability is more than before substitution”. Other than that, the scout bees likewise control the investigation procedure while employee bees and onlooker bee carry out the exploitation process in the search space [85].

The new solutions are randomly searched by the scout bees. The new solution \( x_m \) will be discovered by the scout by using the equation

\[
x_m = l_i + \text{rand}(0,1) \times (u_i - l_i)
\]

 .......... (4.7)

Where, \( \text{rand}(0,1) \) is a random number within the range [0, 1], \( u_i \) and \( l_i \) are the upper and lower bound of the solution space of objective function.

4.7 PSEUDO OF ALGORITHM WITH THE PARAMETERS OF INVENTORY MODEL

Following is the proposed pseudo code of algorithm with the parameters of inventory model:

**Step 1:** “Set the parameters” of artificial bee colony algorithm.

**Step 2:** “Initialize the population of bees” is state for holding cost i.e. case 1 for acceleration coefficient 1.

**Step 3:** “Initialize the population of bees” is state for holding cost i.e. case 2 for acceleration coefficient 2.

**Step 4:** “Find the Abandonment Counter for onlooker bees”.

**Step 5:** For case 1 “check the recruited bees from acceleration coefficient 1”.

**Step 6:** For case 2 “check the recruited bees from acceleration coefficient 2”.

**Step 7:** “Produce the new solution” for employed bees with two cases.

**Step 8:** If \( k \neq i \), then best solution equal to \( \infty \).

**Step 9:** If new bees cost \( \leq \) total population cost the cycle = cycle + 1.

**Step 10:** Now “find the fitness value of all probabilities”.

**Step 11:** “Find the best cost solutions” so far.
“The colony in the ABC algorithm is equally separated into employee bees and onlooker bees. Every arrangement in the search space comprises of an arrangement of optimization parameters”. It implies this colony represented a food source or area for the food source. In this procedure, “every employee bee gets a food source and sent the food source to the onlooker bees”. Toward the finish of this procedure, the quantity of employee bees is equivalent to the quantity of food source.

4.8 CONTROL PARAMETER USED IN BASIC ABC

There are three control parameters utilized as a part of the basic ABC. Briefly they can be described as the quantity of the food sources equivalent to the quantity of “Employee bee (EB)” or “onlooker bee (OL)” required amid “looking food source process (SN)”, the estimation of farthest point and the most maximum cycle number (MCN).

“As indicated by the basic ABC algorithm, the employee bees produce food sources in the nearby area of their present positions. The basic operators used to create neighboring arrangement are insert and swap” [85].

“The insert operator removes a job from its original position j of a permutation π, and then the insert this job into another position k such that (k ∈ (j, j – 1)) whereas the swap operator produces a neighbor by interchanging two jobs of permutation π” [85].

For example, using of common operators swap can be seen in Figure 4.6. “The first table in the Figure 4.6 is the old sequence job before using swap command. The new sequence of job after using swap command is in the second table in Figure 4.6, whereas, Agent 1 has swapped with Agent 3. Position for other Agents do not changed” [85].
**Figure 4.6: Example using of common operators swap**

Figure 4.7 shows the example of sequence that uses insert command to change the position for the food source. “The first table in the Figure 4.7 is the old sequence job before using insert command. The new sequence of job after using insert command is in the second table in Figure 4.7. Position for Agent 5 before insert process is in the last position, after the insert process the position for Agent 5 changes between Agent 1 and Agent 2”.

The new sequence can be seen in the second table in the figure 4.7. Position for other jobs also changed when using insert command.

---

<table>
<thead>
<tr>
<th>“Agent 1”</th>
<th>“Agent 2”</th>
<th>“Agent 3”</th>
<th>“Agent 4”</th>
<th>“Agent 5”</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘Job 1’</td>
<td>‘Job 5’</td>
<td>‘Job 2’</td>
<td>‘Job 6’</td>
<td>‘Job 13’</td>
</tr>
<tr>
<td>‘Job 3’</td>
<td>‘Job 8’</td>
<td>‘Job 12’</td>
<td>‘Job 9’</td>
<td>‘Job 14’</td>
</tr>
<tr>
<td></td>
<td>‘Job 11’</td>
<td>‘Job 4’</td>
<td>‘Job 10’</td>
<td>‘Job 7’</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Agent 3”</th>
<th>“Agent 2”</th>
<th>“Agent 1”</th>
<th>“Agent 4”</th>
<th>“Agent 5”</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘Job 2’</td>
<td>‘Job 5’</td>
<td>‘Job 1’</td>
<td>‘Job 6’</td>
<td>‘Job 13’</td>
</tr>
<tr>
<td>‘Job 12’</td>
<td>‘Job 8’</td>
<td>‘Job 3’</td>
<td>‘Job 9’</td>
<td>‘Job 14’</td>
</tr>
<tr>
<td>‘Job 4’</td>
<td>‘Job 11’</td>
<td></td>
<td>‘Job 10’</td>
<td>‘Job 7’</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>“Agent 1”</th>
<th>“Agent 2”</th>
<th>“Agent 3”</th>
<th>“Agent 4”</th>
<th>“Agent 5”</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘Job 1’</td>
<td>‘Job 5’</td>
<td>‘Job 2’</td>
<td>‘Job 6’</td>
<td>‘Job 13’</td>
</tr>
<tr>
<td>‘Job 3’</td>
<td>‘Job 8’</td>
<td>‘Job 12’</td>
<td>‘Job 9’</td>
<td>‘Job 14’</td>
</tr>
<tr>
<td></td>
<td>‘Job 11’</td>
<td>‘Job 4’</td>
<td>‘Job 10’</td>
<td>‘Job 7’</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Agent 1”</th>
<th>“Agent 5”</th>
<th>“Agent 2”</th>
<th>“Agent 3”</th>
<th>“Agent 4”</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘Job 1’</td>
<td>‘Job 13’</td>
<td>‘Job 5’</td>
<td>‘Job 2’</td>
<td>‘Job 6’</td>
</tr>
<tr>
<td>‘Job 3’</td>
<td>‘Job 14’</td>
<td>‘Job 8’</td>
<td>‘Job 12’</td>
<td>‘Job 9’</td>
</tr>
<tr>
<td></td>
<td>‘Job 7’</td>
<td>‘Job 11’</td>
<td>‘Job 4’</td>
<td>‘Job 10’</td>
</tr>
</tbody>
</table>

**Figure 4.7: Example using of common operators insert**
One advantage of the ABC algorithm is the low number of parameters required, as shown in Table 4.2. The parameter SN is the number of solutions or food source, it is also the number of employed bees and the number of onlooker bees. “MCN is the total number of cycles or iterations of the algorithm. Limit is the number of cycles that a non improved solution kept before being replaced by a new solution generated by the scout bee mechanism. The colony size, the total number of employed bee and onlooker bees is 2 * SN as each group has SN individuals”[85].

Table 4.4: ABC Algorithm Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution Number</td>
<td>SN</td>
<td>Total number of solutions, employed bees and onlooker bees.</td>
</tr>
<tr>
<td>Maximum Cycle Number</td>
<td>MCN</td>
<td>Total number of cycles.</td>
</tr>
<tr>
<td>Limit</td>
<td>Limit</td>
<td>“Number of cycles a non improved solution kept before being replaced by a new solution” generated by the scout bee mechanism.</td>
</tr>
</tbody>
</table>

4.9 PROCEDURE OF ABC ALGORITHM

“The ABC algorithm begins utilizing randomly generated initial population P (G=0) of SN solutions (food source positions). Every solution is a dimensional vector of D (measurement). D implies the quantity of optimization parameters. SN is the solution number for population, which implies the aggregate number of solution i.e. food source”[85].

“Second stage is the point at which the employee bee exploiting the food source. The employee bees send the information to the onlooker bee”. This stage intends to create a candidate solution or a new solution, from the nearby area of old ones [85].

On the off chance that the solution can't be additionally enhanced through a predetermined number of trials limit, the solution is abandoned, and the corresponding employee bee turns into a scout bee. “The onlooker bee picks a food
source depending on the probability estimation of the food source, \( p_i \)”. The following is the formula use to calculate the probability esteem for the food source.

\[
p_{i} = \frac{\text{fit}_i}{\sum_{n=1}^{SN} \text{fit}_n}
\]

Where, “\( \text{fit}_i \) is the fitness value of the solution \( i \) assessed by employee bee which is corresponding to the nectar measure of the food source in the position \( i \)”. \( SN \) in this equation implies the quantity of food source which is equivalent to the quantity of employee bee. “After every solution a new solution is produced and afterward assessed by the artificial bee, the performance comparison with the old food source. In the event that the new food source has equivalent or better nectar over the old food source, it replaces the old food source. Otherwise, the old food source is retained”.

The fitness of every food source \( f(x) \) is controlled by the inverse of its makespan esteem.

The next procedure after initialization is the population of the position or solutions is subjected to repeated cycles of the searching procedure by employed bees, onlooker bees and scout bees. “The employed bees and onlooker bees created an adjustment on the position or solution in her memory to locate a new food source and test the nectar amount i.e. fitness estimation of the new source (new solution)”.

The scout bee search the new food source if the food source got from employed bee and onlooker bee isn’t fulfilled. The scout bee then find the new source.

Karaboga and Basturk (2007) have depicted the methodology for ABC algorithm and all the strategy can be found in Figure 4.8. “The initial phases in this ABC calculation is initialize the population and assess the population. After that, the new solution delivered and assessed the new solution. The decision making process began after the bee assessed the new solution using the greedy selection process. The next step is to calculate the probability esteems for the population. From that point onward, the new solution for the onlooker bees created”. This new solution
created from the solution produced by employee bee and this solution is chosen from the probability esteem. “The new generated solution also needs to assess and applied the greedy selection process. The next step is to decide the abandoned solution for the scout bee, if exists, it replaced with another randomly produced solution by scout bee”. In the event that termination isn't fulfilled, repeat the strategy from second step, otherwise stop the procedure and output is the best food source found up until now.

**Figure 4.8: Procedure of ABC Algorithm**