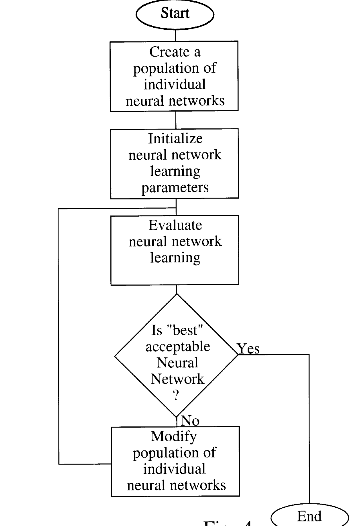
GENETIC ALGORITHM

# **Flowchart**



# **Algorithm**

1. **[Start]** Generate random population of n chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness f(x) of each chromosome x in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
   1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
   2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
   3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
   4. **[Accepting]** Place new offspring in a new population
4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population
6. **[Loop]** Go to step 2

# **Problem statement**

Given the digits 0 through 9 and the operators +, -, \* and /, find a sequence that will represent a given target number. The operators will be applied sequentially from left to right as you read

Example, given the target number 23, the sequence 6+5\*4/2+1 would be one possible solution.

# Stage 1: Encoding

First we need to encode a possible solution as a string of bits i,e a chromosome. So how do we do this? Well, first we need to represent all the different characters available to the solution... that is 0 through 9 and +, -, \* and /. This will represent a gene. Each chromosome will be made up of several genes.

Four bits are required to represent the range of characters used:

0: 0000

1: 0001

2: 0010

3: 0011

4: 0100

5: 0101

6: 0110

7: 0111

8: 1000

9: 1001

+: 1010

-: 1011

\*: 1100

/: 1101

The above show all the different genes required to encode the problem as described. The possible genes 1110 & 1111 will remain unused and will be ignored by the algorithm if encountered.

So now you can see that the solution mentioned above for 23, ' 6+5\*4/2+1' would be represented by nine genes like so:

0110 1010 0101 1100 0100 1101 0010 1010 0001

6 + 5 \* 4 / 2 + 1

These genes are all strung together to form the chromosome:

011010100101110001001101001010100001

# Stage 2: Deciding on a Fitness Function

A fitness score can be assigned that's inversely proportional to the difference between the solution and the value a decoded chromosome represents.

If we assume the target number is 42, making use of the chromosome that is being used as an example

011010100101110001001101001010100001

has a fitness score of 1/(42-23) or 1/19.

As it stands, if a solution is found, a divide by zero error would occur as the fitness would be 1/(42-42). This is not a problem however as we have found what we were looking for... a solution. Therefore a test can be made for this occurrence and the algorithm halted accordingly.

# Stage 3: Running the Code

The code given will parse a chromosome bit string into the values we have discussed and it will attempt to find a solution which uses all the valid symbols it has found. Therefore if the target is 42, + 6 \* 7 / 2 would not give a positive result even though the first four symbols("+ 6 \* 7") do give a valid solution.

# Output

