Hard Computing:

The variables are identified first and classified into two groups as input or conditional variables or antecedents and output variables or action variable or consequents. The input and output variables are expressed in term of the mathematical equation (say differential equation). The differential equations are then solved by analytically or any numerical methods. Control action is decided based on the solution of these mathematical equations.

Main features:

- It yields precise solution. Thus the control action is accurate.
- It is suitable for these problems which are easy to model mathematically and whose stability is highly predictable.

Examples:

1) Traditional numerical optimization methods.
2) Stress analysis by Finite Element Method (FEM).

Soft Computing:

Soft computing methods are developed based on biological approaches or physical science phenomena to solve real life problem where mathematics does not play control role. Soft computing methods have inherited imprecision tolerance and random initial state of the soft computing tools. This introduces a random variability in the model of mathematical systems very similar to random variability which exists in the real system. The algorithm developed based on soft computing may be computationally tractable, robust and adaptive in nature. Soft computing is becomes more and more popular now-a-day in different areas such as optimization intelligent and autonomous robot, pattern recognition, image processing etc.

Most of the real world problems are complex to model mathematically. In such cases we use soft computing methods in which precision is considered to be secondary and are primarily interested to acceptable solution.

Soft computing tools:

Soft computing tools are Fuzzy Logic(FL), Genetic Algorithm(GA), Neural Networks(NN), Ant colony Optimization(ACO), Particle Swarm Optimization(PSO), Simulated Annealing(SA) and so on and two or three combinations of the above tools i.e. FL-GA, GA-NN, GA-FL-NN, SA-GA etc.

Features of soft computing:

- It does not require an extensive mathematical formulation of the problem.
- It may not be able to yield so much precise solution as that obtained by the hard computing methods.
- Different members of this family are Able to perform various types of tasks.
- FL is a powerful technique for dealing with imprecision and uncertainty.

- NN is a potential tool for learning and adaptation.
- GA is an important tool for searching and optimization.
- Algorithm developed based on soft computing is an adaptive in nature. It can be accommodate to the changes of a dynamic environment.

Applications:

- Application of soft computing to handwriting recognition.
- Application of soft computing to automotive systems and manufacturing.
- Application of soft computing to image processing and data compression.
- Application of soft computing to architecture.
- Application of soft computing to decision-support systems.
- Application of soft computing to power systems.
- Neurofuzzy systems.
- Fuzzy logic control.

Hybrid Computing:

Hybrid computing is a combination of the conventional hard computing and the emerging soft computing. Both these computing methods have their inherent advantages and disadvantages. To get the best solution a part of a problem can be solve by hard computing and the remaining part by soft computing. So it is demanded now a day and hybrid computing has been utilized by various investigators.

Drawback of the traditional optimization techniques:

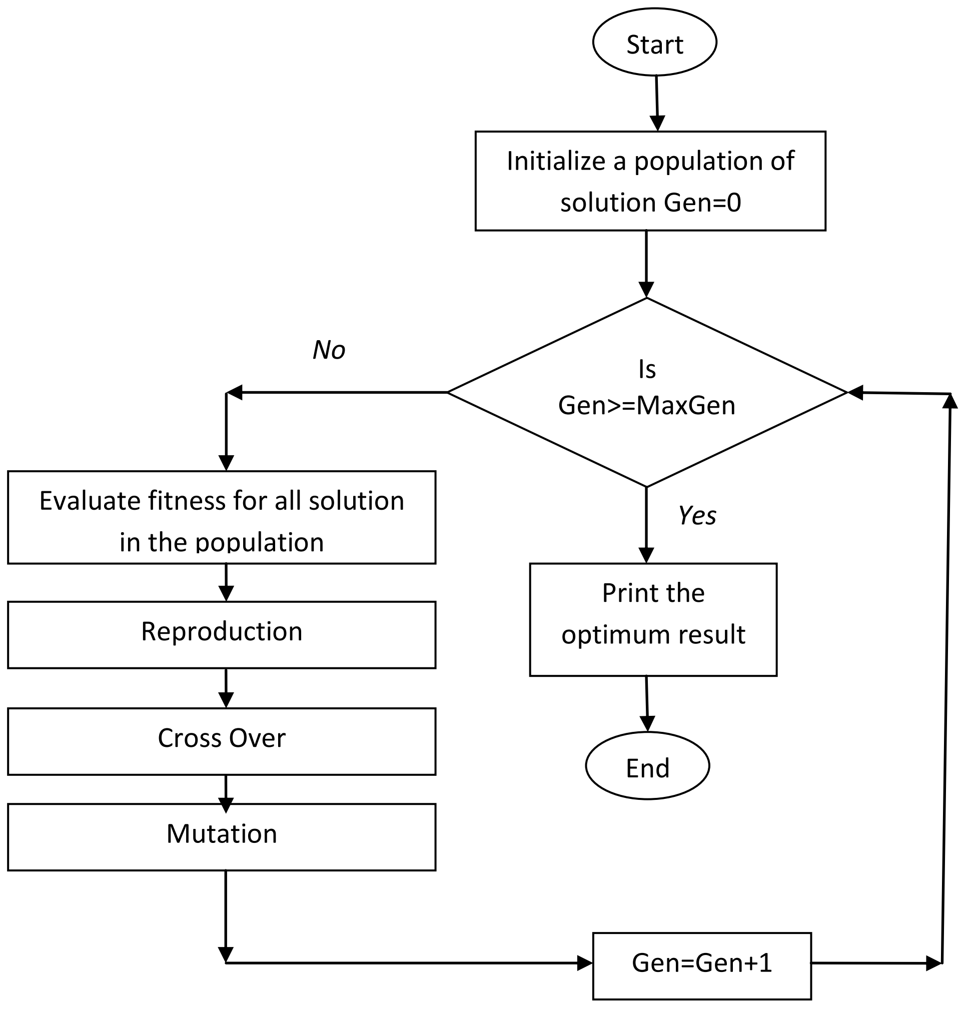
- The final solution depends on the randomly chosen initial solution very often it gets stuck to local optimum.
- For a discontinuous objective function the gradient cannot determine at the point of discontinuous. Hence gradient based method cannot be applied for this type of function.
- These methods may not be suitable for parallel computing.
- These methods are gives only one final optimum solution.

Genetic Algorithm (GA)

Genetic algorithms are computerized population based probabilistic search and optimization. Algorithm based on the mechanism of natural genetics and Darwin’s principle of natural selection i.e. survival of fittest.

Prof. Holland of University of Michigan proposed the concepts of this algorithm in middle of sixties. Several versions of GA are now-a-days available in the literature such as Binary Coded GA, Real Coded GA, Micro GA, Messy GA, Multi Objective GA etc.

Working cycle of GA


Start
Initialize a population of solution Gen=0
Is Gen>=MaxGen
No
Yes
Evaluate fitness for all solution in the population
Reproduction
Cross Over
Mutation
Print the optimum result
End
Gen=Gen+1

Flowchart for GA

Procedure Description:

- ✓ A GA start with a population of initial solution generated at random.
- ✓ The fitness value of the objective function for each solution is calculated. Normally GA works with maximization problem. For minimization problem

$$Min\ f(X) = \begin{cases} Max\ \{-f(X)\} \\ Max\ \frac{1}{f(X)}\ , f(X) \neq 0 \\ Max\ \frac{1}{1+f(X)}\ , f(X) \geq 0 \\ Max\ \frac{1}{1+\{f(X)\}^2}\quad and\ so\ on. \end{cases}$$

✓ The_population solution is modified using stochastic operators-Reproduction, Cross Over, and Mutation.

- **Reproduction:** As all these solution may not be equally good in terms of their fitness values. “Reproduction” is used to select the good solution using their fitness value. It forms as mating pool consisting of good solutions probabilistically. It may be noted that the mating pool may contain multiple copies of a particular good solution. The size of the mating pool is kept equal to that of population of solution considered before reproduction. Thus the average fitness of the mating pool is expected to e higher than that of the pre-reproduction population of solution. Reproduction schemes are
  - Proportional Selection (Roulette Wheel Selection).
  - Ranking Selection.
  - Tournament Selection.

- **Cross Over:** The mating pairs known as parents are selected at random from the above pool which participates in cross-over depending on the value in an exchange of properties between the parents and as results new children are created. Normally if the parents are good children are expected to be good. Different cross over’s are

  1) Single point cross over.
  2) Two points cross over.
  3) Multi-point cross over.
  4) Uniform cross over etc.

- **Mutation:** In biology mutation means a sudden change of parameter on the gene level. In GA it is used for achieving a local change around the current solution. Thus if a solution get stuck at the local optimum. It helps to come out of that and consequently it may jump into the global basin.

  After the above three operations one generation of GA is completed and a new population of solution is obtained. Different criteria are used to terminate the program otherwise the above operations are repeated again.

Binary Coded GA

Let the optimization problem is

$$Max.y = f(x_1, x_2)$$
$$Sub.to \quad x_1^{min} \le x_1 \le x_1^{max}$$
$$x_2^{min} \le x_2 \le x_2^{max}$$

Where x_1 and x_2 are real variables.

Step1: (Generation of population solutions)

An initial population of size N (say N=100, N=150 … depending on the complexity of problem) is selected at random. The solutions are in the form of binary strings composed of 1's and 0's. The length of the binary string is decided based on a desired accuracy in the value of the variables. For example, for accuracy of ε level the accuracy of the string=L (say) $=log_2^{(\frac{x_1^{max}-x_1^{min}}{\varepsilon})}$. Complexity of a binary coded GA =L$\log L$.

The string selected at random are

```
10100 ...1001
01101 ...0101
.....................
01111 ...0011
```

Step2: (Fitness Evaluation)

As x_1 and x_2 are real valued variables. The binary sub-strings assigned to x_1 and x_2 are decoded and corresponding real values are determined as $x_1 = x_1^{min} + \frac{x_1^{max}-x_1^{min}}{2^L-1} \times D.$

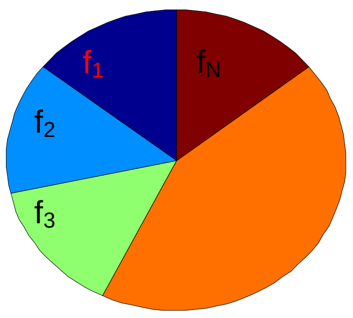
Where D= decoded value of the binary sub-string.

Similarly, $x_2 = x_2^{min} + \frac{x_2^{max}-x_2^{min}}{2^L-1} \times D.$

These values of x_1 and x_2 are put in the function $f(x_1, x_2)$ and the functional value of (x_1, x_2) represents the fitness of the corresponding population.

Step3: (Reproduction (Proportional selection/ Roulette wheel selection))

In this scheme, the probability of a string being selected for the mating pool is considered to be proportional to its fitness. It is implemented with the help of a Roulette-Wheel as shown below



The total surface area of the wheel is divides into N-parts (Where N is the population size) in proportional to the functional values f_1, f_2, $f_3 \ldots f_N$. The wheel is rotated in a particular direction (either clockwise or anti-clockwise) and a fixed pointer is used to indicate the winning area after it stops.

A particular sub-area representing a GA solution is selected to be winner probabilistically. The probability that ith area will be declared and is given by the expression $p=\frac{f_i}{\sum_{i=1}^{N} f_i}$. The wheel is rotated for N-times and each times only one area is identified by the pointer to be the winner. In this process a good string may be selected for a number of times. The procedure is shown below

Code No. Of GA string	**GA string fitness**	**Probability Of being selection**
1	f_1	$\frac{f_1}{F}$
2	f_2	$\frac{f_2}{F}$
3	f_3	$\frac{f_3}{F}$
. . .	. . .	. . .
N	f_N	$\frac{f_N}{F}$

Where $F=\sum_{i=1}^{N} f_i$.

To form the mating pool a sequence of random numbers r_1, $r_2 \ldots$ (Between 0 and 1)(say) are generated. If r_1 is less than or equal to $\frac{f_r}{\sum_{i=1}^{N} f_i}$ then the rth string is selected. Similarly, we selected N-strings for mating pool. In this pool a good string may repeat more than once.

Step4: (Cross Over)

In cross over there is an exchange of properties between the parents and as a result two children solution produced. Here we select the cross over probability say p_c. Then the number of string which will go for cross over is $N \times p_c$.

To select the string the random numbers are generated say r_{11}, r_{12}, $r_{13} \ldots$. If $r_{13} \le p_c$ then the string 3 is selected for cross over. Similarly first $(N \times p_c.)$ strings are selected for cross over out of these string. Strings are selected in pair as parents. There are different types of cross over

1. Single point cross over: We select a cross over site lying between 1 and L-1 at random. where L indicates the string length. Normally left side of the string remains unchanged and the right side is swapped.
   Example, cross over site

$$\text{Parents}\begin{cases}0011 \mid 011110\\ 1100 \mid 101001\end{cases}$$

$$\text{Children}\begin{cases}0011 \mid 101001\\ 1100 \mid 011110\end{cases}$$

2. Two points cross over: We select two different cross over sites lying between 1 and L-1 at random.

Example, cross over sites

$$\text{Parents}\begin{cases}1010 \mid 1110 \mid 111\\ 0110 \mid 1001 \mid 011\end{cases}$$

$$\text{Children}\begin{cases}1010 \mid 1001 \mid 111\\ 0110 \mid 1110 \mid 011\end{cases}$$

3. Multiple Cross over: We select different cross over sites lying between 1 and L-1 at random.
   Example, cross over sites

$$\text{Parents}\begin{cases}101 \mid 111 \mid 111 \mid 001\\ 011 \mid 100 \mid 011 \mid 101\end{cases}$$

$$\text{Children}\begin{cases}101 \mid 100 \mid 111 \mid 101\\ 011 \mid 111 \mid 011 \mid 001\end{cases}$$

4. Uniform cross over: At it bit position of the parent strings we toss a coin if a head appears there will be a swapping of bit among the children strings otherwise they will remain same as parent strings.

Step5: (Mutation)

A mutation probability is selected and normally it is kept to a low value say p_m in the range $(\frac{0.1}{L}, \frac{1}{L})$. To implement a bit-wise mutation scheme a random number lying between (0.0, 1.0) is created at each bit position. If this number $\leq p_m$ at that particular bit position, then that bit will be mutated (i.e. 1 will be changes to 0 and vice verse).

Limitation of Binary coded GA

The main disadvantages of binary coded GA are the following: If we need more precision in the values of variables we have to assign more number of bits to represent them. To ensure a proper search in such a situation population size is kept to high value and as a result of which computational complexity of the GA increases. Thus binary coded GA may not be able to yield any arbitrary precision in the solution.

GA parameters setting

GA parameters are:

1) Population size (N).
2) Cross-over probability (p_c).
3) Mutation probability (p_m).
4) Maximum number of generation/ iterations (MAXGEN).

The performance of the genetic search depends on the amount of exploration (population diversity) and exploitation (selection pressure). To have an effective search there must be a proper balance between them and to ensure this the above GA parameters are to be selected in the optimal sense. The optimal GA parameters are problem dependent i.e. vary from problem to problem. Normally cross-over probability (p_c) is in range (0.6 to 1.0), Mutation probability (p_m) is in range (0.001 to 0.011), Population size (N) is in range (50 to 150), Maximum number of generation/ iterations (MAXGEN) is in range (100 to 250) or (500 to 1000) or (50 to 150).

Constraint optimization problem using GA

A constraint optimization problem (either Max. or Min.) may be expressed as

$$optimize\ f(X).\ .\ \ .\ \ .\ .(1)$$

$$X = (x_1, x_2, x_3 \dots x_n)$$

sub. to

$$g_i(X) \le 0,\ i = 1,2, \dots n\ .\ \ .\ \ .\ .(2)$$

$$h_i(X) \le 0,\ i = 1,2, \dots p\ .\ \ .\ \ .\ .(3)$$

and

$$\begin{aligned} x_1^{min} &\le x_1 \le x_1^{max} \\ x_2^{min} &\le x_2 \le x_2^{max} \\ . \quad . &\quad . \quad . \quad . \quad . \quad . \\ x_n^{min} &\le x_n \le x_n^{max} \end{aligned}$$

There are some number of constraint handling techniques in GA which are

1) Penalty function method/ Approach.
2) Method of maintain a feasible population over infeasible solution.
3) Approach aiming at preserving feasibility of solution.
4) Approach separating the objectives and constraints.

Penalty function method: In this approach the fitness function solution (say ith solution) is expressed by the modifying its objective function as

$$F_i(X) = f_i(X) \pm p_i\ .\ \ .\ \ .\ \ .\ \ .\ \ .\ .(4)$$

Where p_i indicates the penalty used to penalize as infeasible solution. For a feasible solution p_i is set equal to 0.0 as for infeasible solution

$$p_i = c\sum_{k=1}^{q}\{Q_{ik}(X)\}^2.\ \ \ .\ \ .\ \ .\ \ .\ \ .\ (5)$$

where c is the user defined penalty coefficient and $\{Q_{ik}(X)\}^2$are penalty terms for the kth constraint corresponding to ith objective function taking both constraints (2) and (3) into account here q=n+p. The above penalty could be again either static or dynamic or adaptive in nature.

Advantages of GA

1) Normally its gives global optimal solution i.e. the chance of its solution for being trapped into the local minimum is less.
2) They can handle the integer programming problem efficiently.
3) As gradient information of the objective function is not required by a GA. It can optimize discontinuous objective function also.
4) It is suitable for parallel implementation.
5) The same GA with little bit of modification in the string can solve a variety of problem. Thus it is a versatile optimization tool.

Disadvantages of GA

1) It is computationally expensive and consequently has a slow convergence rate.
2) It works like a black-box optimization tool.
3) There is no mathematically convergence proof till-to-day.
4) A use must have a proper knowledge how to select an appropriate set of GA parameters.

Real coded GA

For the real coded GA, several version of cross over and mutation are available.

Cross over operators:

1. **Simple cross over:** It is define as follows:
   If $X_1=(x_1,x_2\ldots x_q)$ and $X_2=(y_1,y_2\ldots y_q)$ are crossed after the kth position the resulting off spring are $X_1'=(x_1,x_2\ldots x_k,y_{k+1},y_{k+2}\ldots y_q)$ and $X_2'=(y_1,y_2\ldots y_k,x_{k+1},x_{k+2}\ldots x_q)$.
   Example,
   $X_1=(x_1, x_2) = (-2.4, 11.5)$
   $X_2=(y_1, y_2) = (4.5, 10.3)$
   $X_1'= (-2.4, 10.3)$ and $X_2'= (4.5, 11.5)$
   But such an operation produces off spring outside of the domain D. To avoid this, we use the property of convex spaces that there exists aε[0,1] such that
   $X_1'=(x_1, x_2\ldots x_k, ay_{k+1}+(1-a)\, y_{k+1},ay_{k+2}+(1-a)\, y_{k+2}\ldots ay_q+(1-a)\, y_q)$
   and $X_2'=(y_1,y_2\ldots y_k, ax_{k+1}+(1-a)\, x_{k+1},ax_{k+2}+(1-a)\, x_{k+2}\ldots ax_q+(1-a)\, x_q)$.
2. **Arithmetic Cross over:** It is define as a linear combination of two vectors. If X_1^t and X_2^t are crossed. The resulting off springs are X_1^{t+1} =a$X_1^t + (1-a)X_2^t$

and X_2^{t+1} = (1-a)$X_1^t + aX_2^t$. Where a is either a constant (uniform arithmetic cross over) or a variable whose value depends on the age of population (non-uniform arithmetic cross over).

Mutation operators:

1. Random mutation: Mutated solution is obtained as $pr_{mutated} = pr_{original} + (r-0.5)\Delta$

Where r is lying between 0.0 and 1.0, Δ is the max value of per mutation defines by the user.

Example,

Let $pr_{original}$=15.6, r=0.7, Δ=2.5

Therefore, $pr_{mutated}$=1.5.6+(0.7-0.56).2.5=16.1

2. Uniform mutation:

This operator requires a single parent x (say) and produce a single off spring x' (say). The operator selects a random component kε{1, 2…q} of a vector x=(x_1, x_2…x_q) and produce x' =(x_1, x_2…x'_k…x_q). Where x'_k is a random value (uniform probability distribution) from the range [x_k^{min} , x_k^{max}].

Example, Max f(x) = $x^3 - 12x^2 + 45x$ using real coded genetic algorithm with 0≤x≤4.

Sol.: Here, range of x is 0≤x≤4 and let the population size =5.

Also given probability of cross over =0.4 & probability of mutation=0.2.

Generate random no. between (0, 4) are 1.852, 3.828, 1.380, 1.472, and 1.776.

Let initial population solution are x_1 =1.852 x_2 =3.828, x_3 =1.380, x_4 =1.472, x_5 =1.776.

Chromosome No.	**Initial Value of x**	**Fitness value**	**Probability**	**Cumulative probability**
1.	1.852	48.53	0.207 $(=\frac{48.53}{234.02})$	0.207
2.	3.828	52.51	0.224	0.431
3.	1.380	41.88	0.179	0.610
4.	1.472	43.43	0.186	0.796
5.	1.776	47.67	0.204	1.000
		Total=234.02		

Generate random no. between (0, 1) are 0.46, 0.30, 0.82, 0.90, and 0.56. Selected chromosomes are x_3, x_2, x_5, x_5, x_3 i.e, after selection population becomes x_1' =1.380, x_2' =3.828, x_3' =1.776, x_4' =1.776, x_5' =1.380.

Chromosome No.	**Value of x**	**Fitness value**
1	3.570	53.98
2	2.224	51.72
3	1.776	47.60
4	1.776	47.60
5	1.380	41.87

Let no. of chromosome will go for cross over =0.4×5=2

Generate random no. between (0, 1) are 0.346, 0.130, 0.982, 0.090, and 0.656.

Chromosomes selected for cross over are x_1' and x_2' . (As 0.346, 0.130 are less than 0.4).

Let λ be a random number between (0, 1) and 0.346 (say). Using arithmetic cross over off-springs are

$$x_1'' = \lambda x_1' + (1-\lambda)x_2' = 2.970$$
$$x_2'' = (1-\lambda)x_1' + \lambda x_2' = 2.224$$

After cross over operation (arithmetic cross over) off-springs are x_1'' =2.970, x_2'' =2.224

Population after cross over are x_1'' =2.970, x_2'' =2.224, x_3' =1.776, x_4' =1.776, x_5' =1.380.

Let no. of chromosomes mutated = 0.2×5=1.

Generate random nos. are 0.19, 0.59, 0.65, 0.45, and 0.96.

Chromosomes to be mutated= x_1'' (As 0.19 less than 0.2).

Using random mutation, where r=0.55 and Δ=1.20.

$pr_{mutated} = pr_{original} + (r-0.5)\Delta$ = 2.970+(0.55-0.5)×1.2 = 3.57.

After mutation operator (random mutation)Population becomes x_1''' =3.570, x_2'' =2.224, x_3' =1.776, x_4' =1.776, x_5' =1.380.

After one iteration population with corresponding fitness values as follows

Here best fitted value = 53.98, which is greater than the best fittest value at the beginning i.e. 52.51.