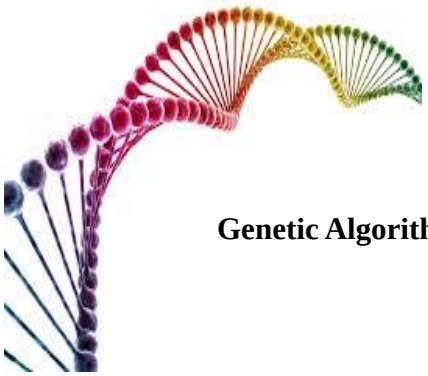


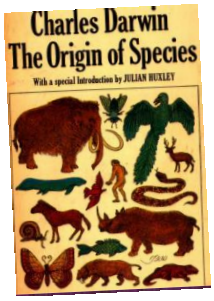
Genetic Algorithm

A General Optimization Problem

Minimize (Maximize) f (X),
where f: $R^n \rightarrow R$, $X = (x_1, x_2, x_3 \ldots x_n)$
s.t. $X \in D \subseteq R^n$, where D is defined by
$g_k(X) \geq 0$, k=1, 2, …… m
$h_j(X) = 0$, j=1, 2, …… p
$a_i \leq x_i \leq b_i$, i=1, 2, …… n

f(X) is the objective function
$X = (x_1, x_2, x_3 \ldots x_n)$ are decision variables
D is feasible domain or search space
$g_k(X)$ are inequality constraints (they are m in number)
$h_j(X)$ are the equality constraints(they are p in number)
a_i and b_i are the lower and upper bounds on the decision variables

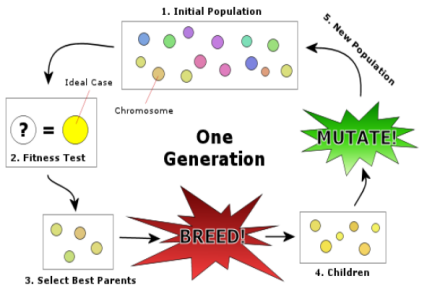
Darwin's Theory of Evolution



Inheritance or the passing of features from one generation to the next.

Competition or survival of the fittest.

One Cycle of GA



GA Terminology

POPULATION
❑Collection of several alternate solutions of the given problem.

CHROMOSOME/ STRING
❑Each individual in the population

❑ GENES
Individual character or symbol in the chromosomes

POPULATION SIZE
❑Number of individuals

FITNESS FUNCTION
❑An evolution function used to determine the fitness of each candidate solution, usually user defined & problem specific.

FITNESS
Value of the fitness function evaluated at each chromosome

Population Initialization

- Loaded from secondary storage
- Randomly generated

Population Size

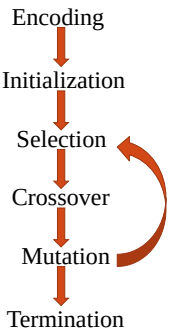
- Small size
  - Less exploration
  - faster convergence (maybe to a local optima)
- Large size
  - Large computational time
  - Large memory requirement

Fitness Function

f(X) is objective function
F(X) is fitness function
For maximization problems F(X) = f(X)
For minimization problems
F(X) = 1/ f(X) if f(X) ≠ 0
F(X) = 1/(1 + f(X)) if f(X) = 0

Or, some other fitness function may also be chosen depending on the problem.

Genetic Algorithm



Encoding

- ❑Binary Encoding
- ❑Real Encoding
- ❑Permutation Encoding
- ❑Octal Encoding
- ❑Hexadecimal Encoding

Binary Encoding

Every chromosome is a string of bits of 0 or 1.

4-bit string	Numeric Value
0000	0
0001	1
0010	2
0011	3
0100	4
…..	…..
1111	15

Permutation Encoding

In **permutation encoding**, every chromosome is a string of numbers, which represents number in a **sequence**.
Ch A = 1 4 5 2 6 3 7 9 8
Ch B = 3 1 7 2 8 9 5 4 6
- Used for ordering problems
- Eg. Travelling Salesman or task ordering problem

Real Encoding

Using the real numbers as it is from the feasible domain to represent the population.

Selection

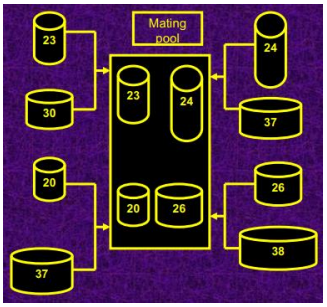
Preference to better individuals, allowing them to pass on their genes to the next generation.

Various selection processes are present like:

- ❑Tournament Selection
- ❑Roulette Wheel Selection
- ❑Bank Selection
- ❑Boltzmann Selection

Tournament selection quiet famous.

Tournament Selection



Example

max f = …
s.t. 0 < x1 , x2 < 90

Ind. No.	String	x_1	x_2	F
1	0000 0000	0	0	1
2	0010 0001	12	6	2.1
3	0001 0101	6	30	3.11
4	0010 1000	12	48	4.01
5	0110 1010	36	60	4.66
6	1110 1000	84	48	1.91
7	1110 1101	84	78	1.93
8	0111 1100	42	72	4.55

Example

Ind. No.	String	x_1	x_2	F
1	0000 0000	0	0	1
2	0010 0001	12	6	2.1
3	0001 0101	6	30	3.11
4	0010 1000	12	48	4.01
5	0110 1010	36	60	4.66
6	1110 1000	84	48	1.91
7	1110 1101	84	78	1.93
8	0111 1100	42	72	4.55

Individuals	Winner	String
2 and 4	4	0010 1000
3 and 8	8	0111 1100
1 and 3	3	0001 0101
4 and 5	5	0110 1010
6 and 1	6	1110 1000
2 and 1	2	0010 0001
4 and 2	4	0010 1000
8 and 3	8	0111 1100

1, 7 are not selected
4, 8 are selected twice
Others selected once

Example

The New Population:

Ind. No.	String	Fitness
1	0010 1000	4.01
2	0111 1100	4.55
3	0001 0101	3.11
4	0110 1010	4.66
5	1110 1000	1.91
6	0010 0001	2.1
7	0010 1000	4.01
8	0111 1100	4.55

Crossover Operators

- ❑Single point crossover

Chromosome 1	10011001 00110110
Chromosome 2	11001110 00011110
Offspring 1	10011110 00011110
Offspring 2	11001001 00110110

❑Two point crossover

Chromosome 1	10011001 00110110
Chromosome 2	11001110 00011010
Offspring 1	10011110 00011110
Offspring 2	11001001 00110010

❑ Cut and Splice Crossover

Chromosome 1	110 | 11001 00110110
Chromosome 2	11011110 00 | 011110
Offspring 1	110011110
Offspring 2	11011110 0011001 00110110

❑ Uniform Crossover

Chromosome 1	11011001 00110110
Chromosome 2	10011110 00011100
Offspring 1	11011000 00010100
Offspring 2	10011111 00111110

❑ Heuristic Crossover

Parents :
$X^1 = (x_1^1, x_2^1, \ldots\ldots\ldots\ldots x_n^1)$
$X^2 = (x_1^2, x_2^2, \ldots\ldots\ldots\ldots x_n^2)$

Offspring:
$Y = (y_1, y_2, \ldots\ldots\ldots\ldots y_n)$
where
$yi = u\,|x_i^2 - x_i^1| + x_i^2$ for i=1, 2, …… n
u is uniformly distributed random number in the interval [0,1]

❑ Arithmetic Crossover

Parents :
$X^1 = (x_1^1, x_2^1, \ldots\ldots\ldots\ldots x_n^1)$
$X^2 = (x_1^2, x_2^2, \ldots\ldots\ldots\ldots x_n^2)$

Offspring:
$Y^1 = (y_1, y_2, \ldots\ldots\ldots\ldots y_n)$
$Y^2 = (y_1, y_2, \ldots\ldots\ldots\ldots y_n)$

where
$y_i^1 = u\,x_i^1 + (1 - u)x_i^2$ for i=1, 2, …… n
$y_i^2 = u\,x_i^2 + (1 - u)x_i^1$ for i=1, 2, …… n
where u is chosen randomly in [0,1]

Probability of Crossover

Def: P_c It is a real number between 0 and 1, which decides the percentage of chromosomes that participate in the crossover.

Ex. If $P_c = 0.25$, then 25 % of the chromosomes of the population participate in the crossover.

Procedure for Crossover

```
begin
    k ← 0;
    while (k < pop_size) do
    r_k ← random number between 0 and 1;
        If (r_k < 0.25) then
            Select chrom_k as one parent for crossover
        endif
    k ←k + 1;
    endwhile
end
```

Mutation

Ex: Let population be:
Ch 1 = 0110 1011
Ch 2 = 0011 1101
Ch 3 = 0001 0110
Ch 4 = 0111 1100

Suppose minima has 1 at the first bit position. Can it be obtained by crossover?

Advantages of Mutation

- Helps escape local minima
- Maintain diversity in population

Single Point Mutation

Original offspring	1101111000011110
Mutated offspring	1100111000011110

Parameters of GA

- Crossover probability
- Mutation probability
- Length of chromosome
- Population size

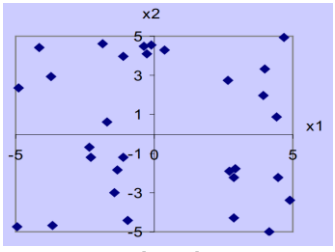
Example

Minimize De Jong Function

$\Rightarrow$Min $\sum_{i=1}^{n} x^2$

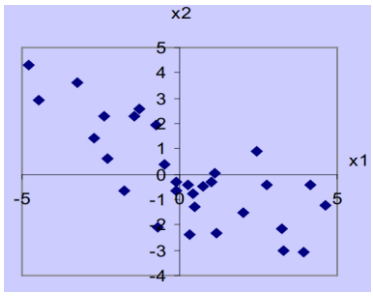
Subject to $-5 \leq x \leq 5$

The minima would lie at (0,0)

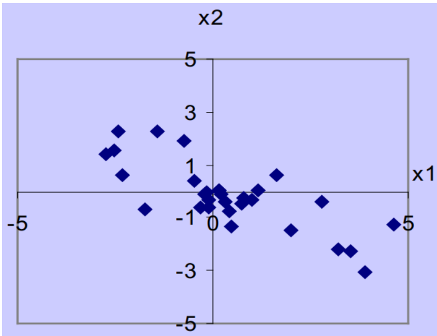


Initial Population

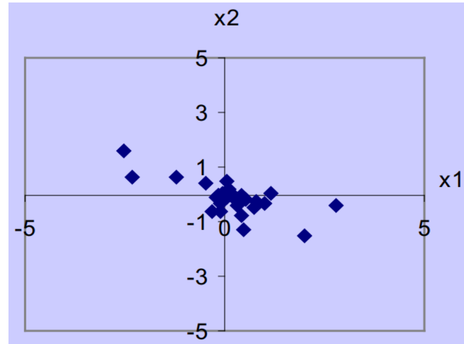
Convergence Over The Iterations



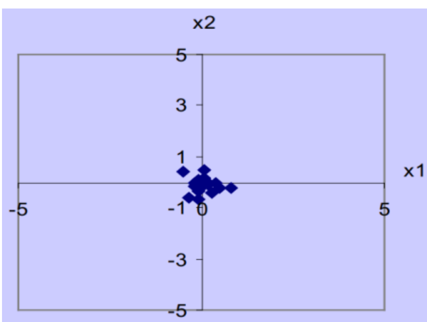
Convergence Over The Iterations



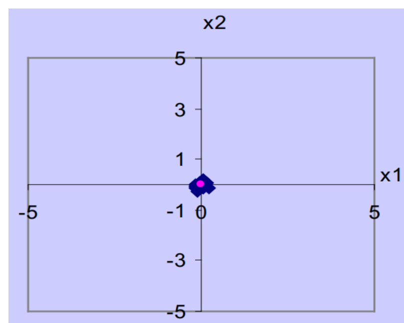
Convergence Over The Iterations



Convergence Over The Iterations



Convergence Over The Iterations



Difference Between GA and Other Traditional Algorithms

- GA works with a coding of Parameter set, not parameters themselves.
- GA searches from a population of solutions, not a single solution.
- GA uses payoff (objective function) information, not derivatives or other auxiliary knowledge.
- GA uses probabilistic transition rules, not deterministic rules.

Applications of GA

- Nonlinear dynamical systems - predicting, data analysis
- Designing neural networks, both architecture and weights
- Robot trajectory
- Evolving LISP programs (genetic programming)
- Strategy planning
- Finding shape of protein molecules
- Scheduling problems
- Game evaluations : Sudoku, etc.
- Music generation
- Structural optimization
- Pattern recognition, etc.