Unit-01

## Evolutionary computing:-

Evolutionary computing draws its inspiration from the process of natural selection, a fundamental mechanism driving biological Evolution.

* Evolutionary computation is a computational intelligence model that mimics biological evolution processes using algorithms like genetic algorithm, Evolutionary programming, evolution strategies and genetic programming
* Evolutionary computing is a field of research that develops problem-solving algorithms inspired by the process of natural evdution.
* Ex. These algorithms, called Evolutionary Algorithms use ideas from Darwin's theory - such as reproduction, mutation and natural selection to search for good solutions to complex problems.

# UNIT-1

## 1) Problem solving as a search task in NIC

In computing, many problems (like optimization, scheduling, pathfinding) can be framed as search tasks.

- The problem = search space (all possible soln's)
- The goal = find the best soln

⇒ In NIC, search is not done by brute force. Instead, NIC uses heuristics inspired by natural processes (evolution, swarm behavior, learning, adaptation) to explore the search space efficiently.

⇒ problem solving as search

① Initial state - where the search begins
② Search space - All possible states/solns
③ Operators / Transitions - Rule for moving b/w states
④ Goal state - Desired / optimal soln
⑤ Evaluation Func" - measures how good a soln is.

Example

Cities A, B, C, D, E = 5

⇒ Search space:- All 5! = 120 possible routes

⇒ ACO process:-

- Ants explore paths randomly
- Over time, stronger pheromone builds on shorter routes
- Final solution

A → C → E → B → D → A

(shortest path)

Ant colony optimization algorithm is used. Travelling sales persun is also the best example for it.

---

2) Hill climbing in NIC

Idea → Hill climbing is a local search optimization algorithm inspired by the natural process of climbing a hill to reach the highest peak.

It assumes that by taking steps that improve the 'fitness' (objective function), one will eventually reach a peak (solution)

(solving the 8-queens problem but may get stuck in a partial sol?) best or

Working -

① start with an initial soln.
② evaluate neighboring solns.
③ Move to the neighbor with a better value.
④ Repeat until no better neighbor exists.

Strengths -

- simple and easy to implement
- Works well for problems with smooth landscapes.

Limitations -

- Gets stuck in local maxima/minima
- Doesn't explore enough to find global solutions.

ex in nature :-

Animals foraging for food may follow the path of increasing scent intensity, climbing toward a peak of food availability - but might stop if they get trapped in a small local food source instead of finding the biggest one

# *Simulated Annealing in NIC

Idea - SA is inspired by the annealing process in metallurgy - where metals are heated and cooled slowly to reduce defects and reach a stable crystalline structure.

Working -

1. Start with a initial solution and a high "temperature".
2. Pick a neighbor solution
3. If the neighbor is better, move there.
4. If it is worse, accept it with some probability $P = e^{-\Delta E/T}$, where T=Temp
5. Gradually reduce temperature
6. End when the system is "Frozen".

Strengths

- Avoids local maxima / minima by allowing controlled random moves
- Good balance between exploration and exploitation.

Limitations - Slower than hill climbing
- Requires careful tuning of cooling schedule.

Example in nature

- similar to metal cooling in physical annealing
- can also be seen in biological adaptation, where sometimes organisms accept a "worse" strategy temporarily (mutation / variation) that may later lead to a better long-term adaptation

*Bester → 8-queens problem, but it can escape dead ends by accepting worse moves occasionally, often finding the global solution

3) Evolutionary Biology (Nature side)

EB studies how living organisms evolve over generations through processes like natural selection, mutation, recombination and survival of the fittest.

Key concepts

population - A group of individuals (organisms)

Fitness - Ability to survive & reproduce

Selection - Fittest individuals are more likely to pass genes

mutation - Random changes in DNA introducing diversity

Crossover / Recombination - Mixing genetic material during reproduction

Evolution - Over time, species adapt better to environments.

*Evolutionary computing (Artificial side)

EC is a subfield of AI & NIC that uses principles of evolutionary biology to solve complex optimization problems.

* Key Algorithms -

Genetic alg (GA)
Gentic programming (GP)
evolution strategies (ES)
differential evolution (DE)

## How it works (analogy with biology)

1. Population – A set of candidate soln
2. Fitness Function – Evaluate how good each soln is
3. Selection – choose better soltion for reproduction
4. Crossover – combine two soln to form a new one
5. Mutation – Introduce random variations
6. Evolution loop – Repeat until the best solution is found.

## Examples :-

⇒ EB – Darwin's theory; Finches with better beak shapes and reproduce in the Galapagos islands.

⇒ EC – Genetic alg optimizing the Traveling salesman problem (TSP) where "best paths" survive and combine to evolve toward the shortest tour.

4) Main evolutionary algorithms (5)

There are 4 main

① Genetic Algorithms (GA)

GA is a search and optimization technique inspired by charles Darwin's theory of natural selection ("survival of the fittest").

→ In NIC GA is one of the most popular evolutionary algorithms, used to find near-optimal soln's for complex prob^ns

→ GA representation will be in bit strings, numbers, etc. i.e., candidate solutions encoded as chromosomes.

→ Operators

Selection → choose parents based on fitness

Crossover → combine parts of two parents

mutation → Randomly alter some genes

→ Applications :- optimization, scheduling, ML, path planning.

## ② Genetic programming (GP)

GP is an evolutionary algorithm inspired by Darwinian evolution but instead of evolving fixed-length strings (like in GA), it evolves computer programs or symbolic expressions.

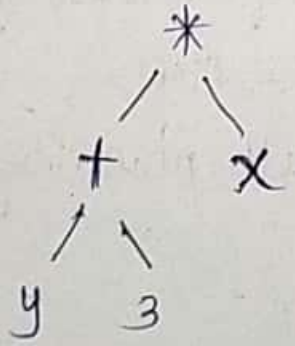
→ It is a specialized branch of evolutionary computing where the goal is to automatically generate programs that solve a problem.

**Representation in GP:** programs are usually represented as tree structures.

⇒ Internal nodes – functions/operators (+, −, *, if-else, sin, cos).

⇒ Leaves (terminals) – Variables, constants, or inputs

Ex:-




→ Represents the program: $(y+3) * x$

Applications -

- Symbolic Regression
- Robotics
- Game AI
- Automatic program Generation

Advantages -

- can generate human-readable programs
- Flexible i.e., works with symbolic data numbers, logics.

operators :

→ crossover - swap subtrees of programs
→ Mutation - Modify parts of a tree

⑤ Evolution strategies (ES)

ES are a type of evolutionary algorithm that originated in germany in 1960s.

- Unlike Genetic algorithms (GA), which often use bit-strings, ES works mainly with real-valued vectors & emphasizes mutation with self-adaptive strategy parameters.
- Goal → optimize continous and high-dimensional problems.

## Representation

- Each individual is a vector of real numbers

eg:- $x = (x_1, x_2, \ldots x_n)$ (solution variables)

Each variable has a associated strategy parameter (mutation strength, $\sigma$)

Finally represented in, $(x_1, \ldots x_n ; \sigma_1, \ldots \sigma_n)$

* Mutation $\Rightarrow$ $x' = x + \sigma \underbrace{N(0,1)}_{\text{normal distribution}}$

## Applications:-

- continuous optimization
- Robotics
- neural network training

## operators:-

- Mutation (Gaussian noise)
- Recombination
- selection ($\mu + \lambda$ strategy or $\mu, \lambda$ strategy)

## (A) Differential Evolution (DE)

DE is a population-based evolutionary algorithm introduced by ston and price in 1995.

→ It is mainly designed for continous optimization problems and is well-known for being simple, fast, and effective.

→ Inspired by the idea of evolution through variations (mutations) caused by differences in traits among individuals in a population.

→ Each solution is represented as a real-valued vector

$$X_i = (x_{i1}, x_{i2}, \cdots x_{iD})$$

where, D = no. of decision variables

Applications - Global optimization
- Engineering design
- ML

operators - mutation (~~Add weighted differ~~
- crossover
- selection //

(5) Evolutionary programming (EP)

EP is inspired from evolution of behavioral strategies.

⇒ Representation - Finite state machines or real-valued vectors

⇒ Operators
- mutation as the primary operator
- No crossover (unlike GA)

⇒ Applications
- optimization
- prediction
- adaptive control

---

5) Scope of Evolutionary computing

EC is a branch of NIC that applies the principles of biological evolution (natural selection, mutation, recombination, survival of the fittest) to solve computational problems

⇒ It includes
- Genetic alg
- Gentic prog
- evolu strategies
- Differential evolu
- evolutia progr

## Scope in problem-solving

Evolutionary computing is powerful because it can handle complex, nonlinear, high-dimensional problems that traditional methods struggle with

→ Optimization problems
- Continous optimization (engineering design, control)
- combinatorial optimization (scheduling routing)

→ Machine learning
- Feature selection, hyperparameter tuning; neural network training
- Automated program/code generation

→ Robotics & control
- Robot path planning, adaptive control systems.

⇒ Data mining & knowledge Discovery
- clustering, classification, rule extraction

⇒ Bioinformatics
- protein folding, gene selection

⇒ Engineering Applications.
- Antenna design, circuit optimization

Why scope is Growing (Advantages)
- Works in black-box scenarios (no gradient/derivative needed)
- Can adapt to dynamic environments
- Good at escaping local optima.
- Flexible representation

Research & Future directions
→ Hybrid algorithms - combining the swarm intelligence & neural networks.

→ Neuroevolution - Deeplearning without backpropagation

- Automated ML → Used to evolve models, architectures and hyperparameters automatically
- Large-scale problems → parallel and distributed EC for big data optimization

- **Biology → Computing Mapping:**

| Biology (Natural Evolution) | Computing (Evolutionary Computing in NIC) |
|---|---|
| Organism / Species | Candidate solution |
| Gene / DNA | Encoded representation (chromosome, vector, tree) |
| Population of organisms | Set of possible solutions |
| Fitness (survival ability) | Objective function value |
| Mutation | Random modification of solution |
| Recombination (crossover) | Combining solutions |
| Natural selection | Choosing best solutions |
| Generations (evolution over time) | Iterations of algorithm |