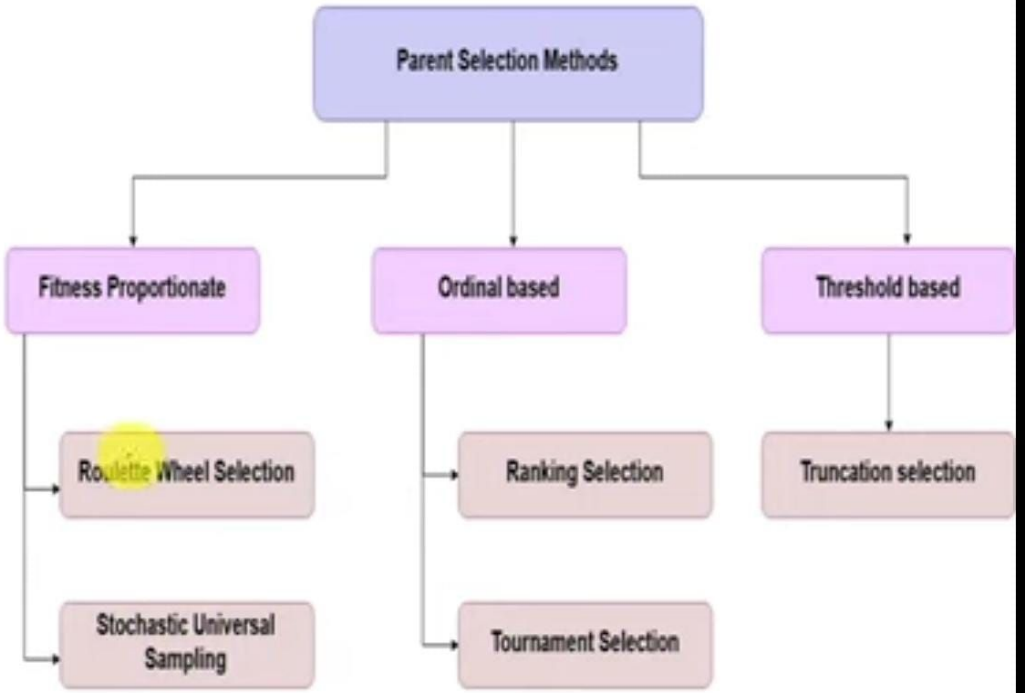
Parent Selection Operators - Genetic Algorithm

Parent selection is a process of choosing a set of chromosomes as a parent for the next generation of the population, based on its fitness value, threshold etc.

Following are the types of parent selection in Genetic Algorithm



Fitness Proportionate Selection Operators

Roulette Wheel Selection

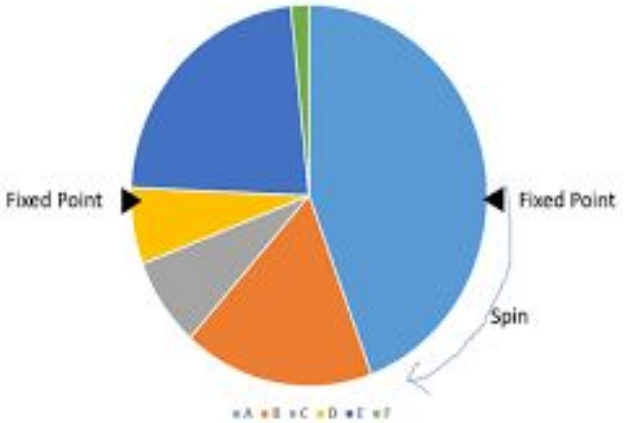
- Parents are selected according to their fitness.
- The better the chromosomes are, the more chances to be selected they have.

Roulette Wheel Selection – Steps

- [Sum] Calculate sum of all chromosome fitnesses in population - sum S.
- [Select] Generate random number from interval (0, S) - r.
- [Loop] Go through the population and sum fitnesses from 0 → sum s.
  When the sum s is greater than r, stop and return the chromosome where you are.

Stochastic Universal Sampling (SUS)

Stochastic Universal Sampling is quite similar to Roulette wheel selection, however instead of having just one fixed point, we have multiple fixed points



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Ordinal-based Selection Operators

Ranking Selection

- In Ranking Selection, we remove the concept of a fitness value while selecting a parent.
- However, every individual in the population is ranked according to their fitness.
- The selection of the parents depends on the rank of each individual and not the fitness.
- The higher ranked individuals are preferred more than the lower ranked ones.

Tournament Selection

- In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent.
- The same process is repeated for selecting the next parent.

Threshold-based Selection Operators

Truncation Selection

- It is an artificial selection method which is used for large populations / mass selection.
- In truncation selection, individuals are sorted according to their fitness.
- Only the best individuals are selected as parents.
- Truncation threshold Trunc is used as the parameter for truncation selection.
- Trunc indicates the proportion of the population to be selected as parents and takes values ranging from 10%–50%.

Encoding and Genetic Operators in Genetic Algorithms

- **Definition**
  Encoding is the process of representing a solution (chromosome) in a form that a computer can process.

- Each encoding method defines how genes represent parameters of the problem

Main types of Encoding

1. Binary Encoding

- **Definition:** It is the most common way of encoding, where each **chromosome** is a **binary string**, which is a solution but not necessarily a best solution. The length of the string depends on the accuracy.
- **Example (Chromosome of binary strings of 16 bits):**
  - `1101 0001 1010 1100`
  - `0111 1001 0011 1001`

2. Octal Encoding

- **Definition: Octal encoding** uses strings made up of **octal numbers** (0–7).
- **Example (Octal encoding of chromosome with eight octal digits):**
  - `53467201`
  - `6135720` (Note: This example appears to have 7 digits, not 8).

Main types of Encoding

3. Hexadecimal Encoding

- **Definition: Hexadecimal encoding** uses strings made up of **hexadecimal numbers** (0–9, A–F).
- **Example (Hexadecimal encoding with size 4):**
  - `9CE7`
  - `3DBA`

4. Permutation Encoding (Real Number Coding)

- **Definition: Permutation encoding** is useful for **ordering problems**. In permutation encoding, every chromosome is a string of **integer/real values**, which represents a number in a sequence.
- **Example (Permutation encoding are as follows):**
  - `153264798; 856723149`
  - `12.5 3.5 10.8 23.5; 10.8 3.5 12.5 23.5`

Main types of Encoding

5. Value Encoding

- **Definition:** In this type of encoding scheme, a **chromosome** is a sequence of values related to the problems. **Value encoding** is very good for some special problems but for this, encoding is often required to develop some new crossover and mutation, specific to the problem. **Value encoding** can be used in problems, where some complicated values are used. Values can be anything from **numbers, real numbers or chars to some complicated objects.**
- **Example (Value encoding chromosomes are as follows):**
  - `1.2324 5.3243 0.4556 2.3293 2.4545`
  - `ABDJEIFJDHDIERJFDLDFLFEGT`
  - `back, back, right, forward, left`

Main types of Encoding

6. Tree Encoding

- **Definition:** This encoding is useful for **program expressions** for **genetic programming**. Each chromosome is a **tree** of some objects such as **functions and commands** of a programming language.

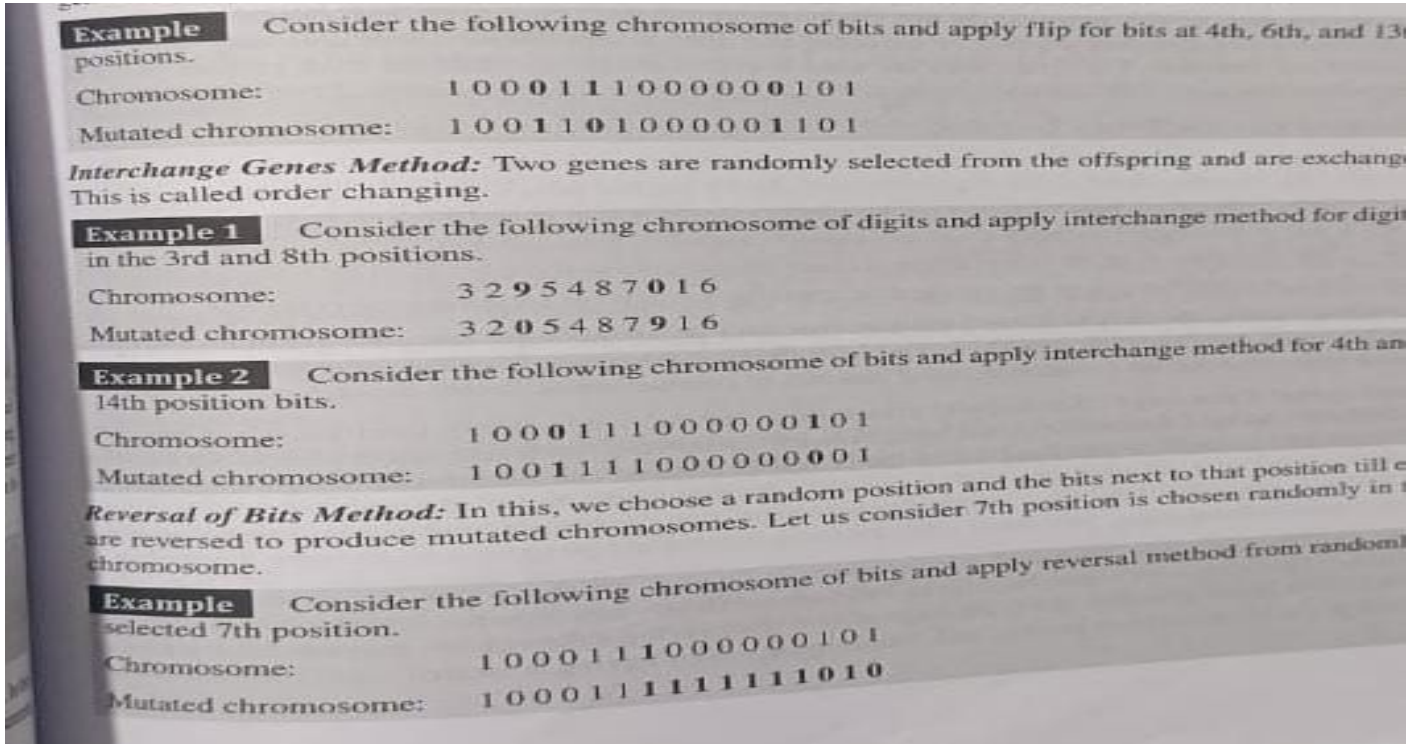
Mutation Methods

Definition and Purpose

Mutation is a simple search operator that alters some of the **chromosomes** (potential solutions). It's important for:

- Getting back the **lost genes** (information).
- Helping to **prevent all solutions in the population from falling into a local optimum**.
- It takes place after the crossover is performed.
- The random changes in genes introduce the **traits** that weren't originally present.
- It helps in the **examination of the entire search space** and **modifies some of the bits** of the building blocks.

Mutation Methods



Example Consider the following chromosome of bits and apply flip for bits at 4th, 6th, and 13
positions.

Chromosome: 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1

Mutated chromosome: 1 0 0 1 1 0 1 0 0 0 0 0 0 1 1 0 1

Interchange Genes Method: Two genes are randomly selected from the offspring and are exchang
This is called order changing.

Example 1 Consider the following chromosome of digits and apply interchange method for digi
in the 3rd and 8th positions.

Chromosome: 3 2 9 5 4 8 7 0 1 6

Mutated chromosome: 3 2 0 5 4 8 7 9 1 6

Example 2 Consider the following chromosome of bits and apply interchange method for 4th an
14th position bits.

Chromosome: 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1

Mutated chromosome: 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1

Reversal of Bits Method: In this, we choose a random position and the bits next to that position till e
are reversed to produce mutated chromosomes. Let us consider 7th position is chosen randomly in
chromosome.

Example Consider the following chromosome of bits and apply reversal method from randoml
selected 7th position.

Chromosome: 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1

Mutated chromosome: 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 0

What is Crossover?

- **Definition**
  Crossover is a genetic operator used to combine genetic information of two or more parents to produce new offspring.
  It mimics natural reproduction and helps in exploring the search space efficiently.

  Purpose
  To generate new solutions (children) with characteristics from both parents.
  Increases diversity and improves the chance of better offspring.

Binary Encoding in GA

- **Binary Representation**
  Chromosomes are strings of 0s and 1s.
  Each bit (gene) represents a specific feature or decision.

  Example:
  Parent 1: 11001010
  Parent 2: 10110100

Types of Crossover (Overview)

- **Main Types under Binary Encoding**
  1. Single Point Crossover
  2. Two Point Crossover
  3. Multi Point Crossover
  4. Uniform Crossover
  5. Arithmetic Crossover
  6. Three Parent Crossover

Single Point Crossover

- **Concept**
  A single crossover point is chosen randomly.
  Bits after that point are swapped between parents.

  Example
  Parent 1: 11001010
  Parent 2: 10110100
  Crossover point: after 4th bit

  Result:
  Child 1: 11000100
  Child 2: 10111010

  Key Point: Simple and preserves large gene blocks.

Two Point Crossover

- **Concept**
  Two crossover points are chosen.
  The section between them is exchanged.

  Example
  Parent 1: 11001010
  Parent 2: 10110100
  Crossover points: after 2nd and 6th bits

  Result:
  Child 1: 11110110
  Child 2: 10001000

  Key Point: Allows more genetic mixing than single point.

Multi-Point Crossover

- **Concept**
  More than two crossover points are used.
  Segments are alternately swapped between parents.

  Example (3-point crossover)
  Parent 1: 11001010
  Parent 2: 10110100
  Crossover points: after 2nd, 4th, and 6th bits

  Result:
  Child 1: 11111000
  Child 2: 10000110

  Key Point: Produces high diversity but can break useful combinations.

Uniform Crossover

- **Concept**
  Each bit in the child is chosen randomly from either parent.
  A random mask decides which parent contributes each bit.

  Example
  Parent 1: 11001010
  Parent 2: 10110100
  Mask: 11001100

  Result:
  Child: 11110100

  Key Point: Very high mixing, maintains diversity.

Arithmetic Crossover

- **Concept**
  Common in real-coded GAs, but can be adapted for binary using bitwise logic (AND, OR, XOR).
  Creates offspring that are logical combinations of parents.

  Example
  Parent 1: 11001010
  Parent 2: 10110100
  Child 1 = AND → 10000000
  Child 2 = OR → 11111110

  Key Point: Maintains logical stability between parents.

Three Parent Crossover

- **Concept**
  Involves three parents.
  Each bit of the child is decided using a majority rule among parents.

  Example
  Parent 1: 11001010
  Parent 2: 10110100
  Parent 3: 11100010

  Result (majority bits):
  Child: 11100010

  Key Point: Produces more stable and balanced offspring.

Classification of Genetic Algorithms

- Genetic Algorithms (GAs) are powerful optimization techniques inspired by natural evolution. They can be classified based on their structure, execution style, and enhancements to improve performance. The main classifications are: **Simple GA, Parallel/Distributed GA, Hybrid GA, Adaptive GA, and Fast Messy GA.**

Classification of Genetic Algorithms

- Genetic Algorithms (GAs) are powerful optimization techniques inspired by natural evolution. They can be classified based on their structure, execution style, and enhancements to improve performance. The main classifications are: **Simple GA, Parallel/Distributed GA, Hybrid GA, Adaptive GA, and Fast Messy GA.**

Classification of Genetic Algorithms

Simple Genetic Algorithm (SGA)

Definition:
A Simple Genetic Algorithm (SGA) is the basic form of GA that performs search and optimization using the objective function associated with each solution (chromosome). SGAs do not require any extra domain knowledge, heuristics, or complex mathematical modeling.

Characteristics:

- Works efficiently on **large, complex, or poorly known search spaces**.
- Useful when **domain knowledge is limited** or hard to encode.
- Does not require **mathematical analysis** of the problem.
- **Easy to implement** and robust compared to traditional optimization methods.

Classification of Genetic Algorithms

2. <u>Parallel and Distributed Genetic Algorithm (PGA)</u>

Definition:

Parallel Genetic Algorithms execute multiple SGAs **simultaneously on different processors** to improve performance and scalability. They are especially useful for **large search spaces** where single SGAs are slow.

- Reduce execution time.
- Avoid premature convergence.
- Improve solution quality by maintaining diversity.

Key Components of PGA:

- Design of **fitness functions**.
- Selection procedures.
- Single or multiple **sub-populations**.
- Rules for **exchanging individuals** between sub-populations.

Classification of Genetic Algorithms

Types of Parallel GA:

1. **Independent PGA:**
    - Each processor has its own population.
    - Populations evolve independently without interaction.
    - Reduces risk of all populations converging to the same poor solution.
2. **Migration PGA:**
    - Extends independent PGA with **periodic migration** of chromosomes.
    - Best chromosomes from one processor replace the worst in another.
    - Prevents early convergence and improves solution quality.
3. **Partition PGA:**
    - Each processor works on a **different portion of the search space**.
    - Helps in exploring diverse areas of the solution space.

Classification of Genetic Algorithms

3. Segmentation PGA:

- Commonly used in problems like the Traveling Salesman Problem (TSP).
- Each processor works on a segment of the chromosome (sub-tour).
- Sub-tours are later combined to form the full solution.

4. Segmentation-Migration PGA:

- Combines the benefits of segmentation and migration.
- Segments are optimized locally, and good solutions are shared periodically
  -

Classification of Genetic Algorithms

3. <u>Hybrid Genetic Algorithm (HGA)</u>

Definition:

A Hybrid GA combines the **global search ability of GAs** with the **speed of local optimization techniques** (like heuristics or local search).

Characteristics:

- Uses GA for broad exploration and local optimization for fine-tuning.
- Can combine specialized crossover operators with heuristic methods.

Advantage:

- Faster convergence than pure GA.
- Higher solution quality for complex optimization problems.

Classification of Genetic Algorithms

4. <u>Adaptive Genetic Algorithm (AGA)</u>

Definition:

Adaptive GAs dynamically adjust their **control parameters** like population size, crossover rate, and mutation probability based on the evolution progress.

Mechanism:

- If the population stagnates (no improvement), **mutation rate increases** to explore new solutions.
- If the population improves steadily, **mutation rate decreases** to focus on exploitation.

Usefulness:

- Optimizes **multi-modal functions** with multiple peaks.
- Helps avoid premature convergence.

Classification of Genetic Algorithms

5. <u>Fast Messy Genetic Algorithm (FMGA)</u>

Definition:

Fast Messy GA is an advanced GA approach that uses **binary strings of variable length** and is capable of **managing building blocks of genetic material** efficiently.

Characteristics:

- Handles **complex, epistatic problems** (where genes interact non-linearly).
- More complex than SGA but can **find high-quality solutions faster**.
- Focuses on preserving and recombining **useful gene segments**.

Applications:

- Problems requiring recognition of **critical gene patterns**.
- Optimization where standard GAs struggle to maintain building blocks.