Assignment : 1

Implement fuzzy set operations.

Union

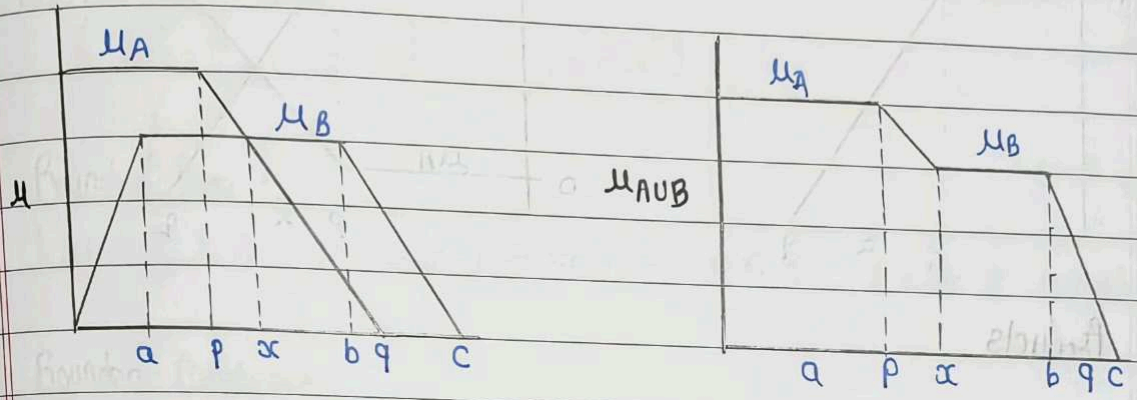
Union (A ∪ B) : $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$

Example :

$A = \{(x_1, 0.5), (x_2, 0.1), (x_3, 0.4)\}$ and

$B = \{(x_1, 0.2), (x_2, 0.3), (x_3, 0.5)\}$;

$C = A \cup B = \{(x_1, 0.5), (x_2, 0.3), (x_3, 0.5)\}$



2. Intersection

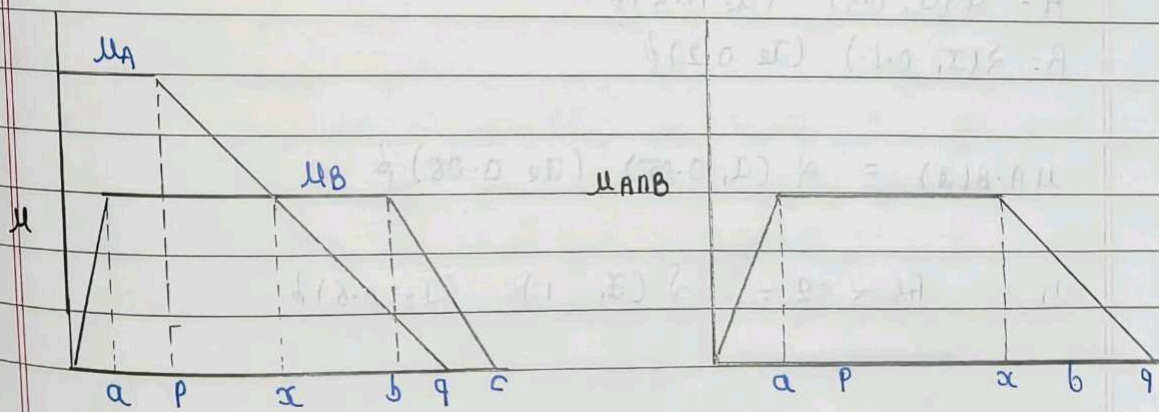
A ∩ B : $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$

Example :

$A = \{(x_1, 0.5), (x_2, 0.1), (x_3, 0.4)\}$ and

$B = \{(x_1, 0.2), (x_2, 0.3), (x_3, 0.5)\}$;

$C = A \cap B = \{(x_1, 0.2), (x_2, 0.1), (x_3, 0.4)\}$



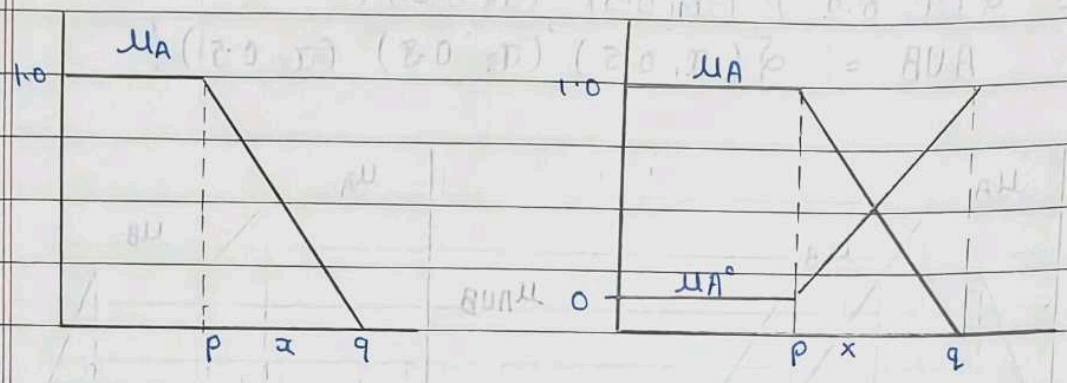
3. Complement

Complement (A^c): $\mu_A^c(x) = 1 - \mu_A(x)$

Example:

$A = \{(x_1, 0.5), (x_2, 0.1), (x_3, 0.4)\}$

$C = A^c = \{(x_1, 0.5), (x_2, 0.9), (x_3, 0.6)\}$



4) Products

Algebric Product of vector product $(A \cdot B)$:

$\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x)$

Scalar Product $(\alpha \times A)$

$\mu_{\alpha A}(x) = \alpha \times \mu_A(x)$

Example

$A = \{(x_1, 0.5), (x_2, 0.3)\}$

$B = \{(x_1, 0.1), (x_2, 0.2)\}$

$\mu_{A \cdot B}(x) = \{(x_1, 0.05), (x_2, 0.06)\}$

$\mu_{\alpha A}$, for $\alpha = 2$ = $\{(x_1, 1), (x_2, 0.6)\}$

Sum & Difference.

Sum (A+B)

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$$

Difference (A-B) $= (A \cap B^c)$

$$\mu_{A-B}(x) = \mu_{A \cap B^c}(x)$$

Disjunctive Sum :

$$A \oplus B = (A^c \cap B) \cup (A \cap B^c)$$

Bounded Sum :

$$|A(x) \oplus B(x)| = \mu|A(x) \oplus B(x)| = \min\{1, \mu_A(x) + \mu_B(x)\}$$

Bounded Difference

$$|A(x) \ominus B(x)| = \mu|A(x) \ominus B(x)| = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$$

Assignment No : 2

* Implement fuzzy relational operations.

1) Crisp relations

Order Pairs:

Suppose A&B are two crisp sets. Then Cartesian product denoted as AxB is a collection of order pairs, such that

$$A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$$

Example:

$A = \{1, 2, 3, 4\}$ $B = \{3, 5, 7\}$

$A \times B = \{(1,3)\ (1,5)\ (1,7)\ (2,3)\ (2,5)\ (2,7)\ (3,3),$
$(3,5)\ (3,7)\ (4,3)\ (4,5)\ (4,7)\}$

Let us define a relation as $R = \{(a,b) \mid b = a+1, (a,b) \in A \times B\}$

Then $R = \{(2,3), (4,5)\}$ in this case.

2) Operations on Crisp relations

- Union : $R(x,y) \cup S(x,y) = \max(R(x,y), S(x,y))$
- Intersection : $R(x,y) \cap S(x,y) = \min(R(x,y), S(x,y))$
- Complement : $\overline{R(x,y)} = 1 - R(x,y)$

3) Composition of two Crisp relations

Given R is a relation on X,Y & S is another relation on Y,Z. Then, RoS is called a composition of relation on X & Z which is defined as follows.

$$RoS = \{(x,z) \mid (x,y) \in R \ \& \ (y,z) \in S \ \& \ \forall y \in Y\}$$

Max - Min Composition

Given the two relation matrices R & S, the max min composition is defined as $T = R \circ S$.

$$T(x,z) = \max_y \{\min\{R(x,y), S(y,z) \;\&\; \forall y \in Y\}\}$$

Example:

Given $X = \{1,3,5\}$; $Y = \{1,3,5\}$; $R\{(x,y) / y = x+2\}$;
$S = \{(x,y) \mid x<y\}$

Here, R & S in on $X \times Y$

Thus we have

$R = \{(1,3), (3,5)\}$, $S = \{(1,3), (1,5), (3,5)\}$

$$R = \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 3 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \quad \text{and} \quad S = \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 3 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

Using max-min Composition

$$R \circ S = \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 3 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

To obtain (1, 1)

we take first row first column

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$$

Take min

$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Take max

$= 0$

4) Fuzzy Relations.

- Fuzzy relation is a fuzzy set defined on the cartesian product of crisp set X_1, X_2, X_n
- Here, n-tuples $(x_1, x_2, \ldots x_n)$ may have varying degree of memberships within the relationship.
- The membership values indicate the strength of the relation between the tuples.

Example.

$X = \{typhoid, viral, cold\}$, $Y = \{running\ nose, high\ temp, shivering\}$

The fuzzy relation R is defined as

R =	running nose	high temp	shivering
typhoid	0.1	0.9	0.8
viral	0.2	0.9	0.7
cold	0.9	0.4	0.6

5) Fuzzy Cartesian Product

Suppose

- A is a fuzzy set on the universe of discourse X with $\mu_A(x) | x \in X$
- B is a fuzzy set on the universe of discourse Y with $\mu_B(y) | y \in Y$.

Then $R = A \times B \subset X \times Y$; where R has its membership function given by $\mu_R(x,y) = \mu_{A \times B}(x,y) = \min\{\mu_A(x), \mu_B(y)\}$

Let R & S be two fuzzy relations on $A \times B$

Union $\mu_{R \cup S}(a,b) = \max\{\mu_R(a,b), \mu_S(a,b)\}$

Intersection $\mu_{R \cap S}(a,b) = \min\{\mu_R(a,b), \mu_S(a,b)\}$

Complement $\bar{\mu}_R(a,b) = 1 - \mu_R(a,b)$

Composition : $T = R \circ S$

$\mu_{R \circ S} = \max_{y \in Y}\{\min(\mu_R(x,y), \mu_S(y,z))\}$

Example :

$X = (x_1, x_2, x_3)$, $Y = (y_1, y_2)$, $Z = (z_1, z_2, z_3)$

$$R = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cc} y_1 & y_2 \end{array} \\ \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.9 \\ 0.8 & 0.6 \end{bmatrix} \end{array} \quad \text{and} \quad S = \begin{array}{c} \\ y_1 \\ y_2 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 0.6 & 0.4 & 0.7 \\ 0.5 & 0.8 & 0.9 \end{bmatrix} \end{array}$$

$$R \circ S = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 0.5 & 0.4 & 0.5 \\ 0.5 & 0.8 & 0.9 \\ 0.6 & 0.6 & 0.7 \end{bmatrix} \end{array}$$

$\mu_{R \circ S}(x_1, y_1) = \max\{\min(\mu_R(x_1, y_1), \mu_S(y_1, z_1)), \min(\mu_R(x_2, y_2), \mu_S$

$= \max\{\min(0.5, 0.6), \min(0.1, 0.5)\}$

$= \max\{0.5, 0.1\}$

$= 0.5$ & so on

Assignment No : 3

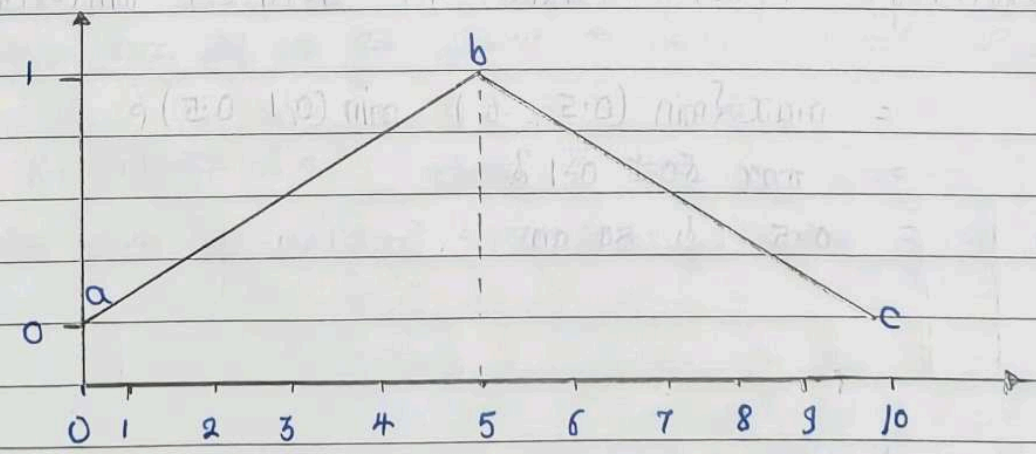
* Implement Any 5 Membeeship Functions

1) Teiangulab

- Shape : A Teiangle
- Pabameteebs : (a, b, c)
  - a = left Point (wheee function statts)
  - b = Peak wheee membeeship = 1
  - c = eight point (wheee Function enas).
- Foemula :

$$\mu(x) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a < x < b \\ \frac{c-x}{c-b} & b \le x < c \\ 0 & x \ge c \end{cases}$$

- Usage: Simple appeoximation fae fuzzy sets like ("medium temp")
- Example: Teiangulae (0,5,10) → eises feom 0 at 0. peaks at 5, falls back to 0 at 10.

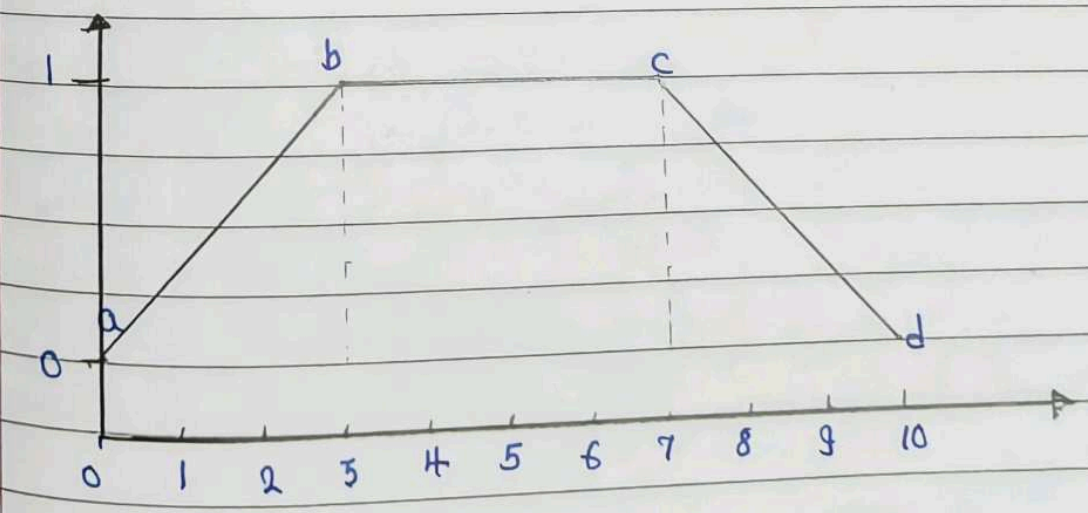


2) Trapezoidal

- shape : A trapezoid (flat top)
- Parameters : (a, b, c, d)
  - a = left start
  - b = start of flat top
  - c = end of flat top
  - d = right end.
- Formula:

$$\mu(x) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a < x < b \\ 1 & b \le x \le c \\ \frac{d-x}{d-c} & c < x < d \\ 0 & x \ge d \end{cases}$$

- usage : Often used for linguistic terms like "low", "high" where range of full membership exists.
- Example : Trapezoidal (0,3,7,10) → slope up, flat top 3-7, slop down

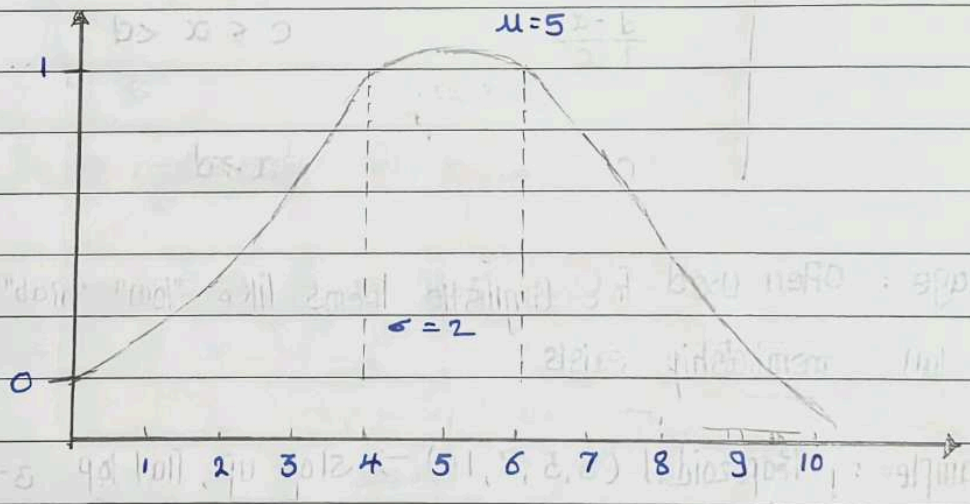


3. Gaussian Membership Function

- shape : Bell Curve
- Parameters :
  - mean (μ): Center (Peak at 1) (shifts curve left / right)
  - sigma (σ): spread (controls width).
- Formula :

$$\mu(x) = e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

- Usage : Smooth & natural (good for real world data uncertainty modelin
- Example : Gausian ($\mu=5$, $\sigma=2$) → peak at 5, spreads smoothly.



Sigmoid

shape : S-shaped curve

Parameters :

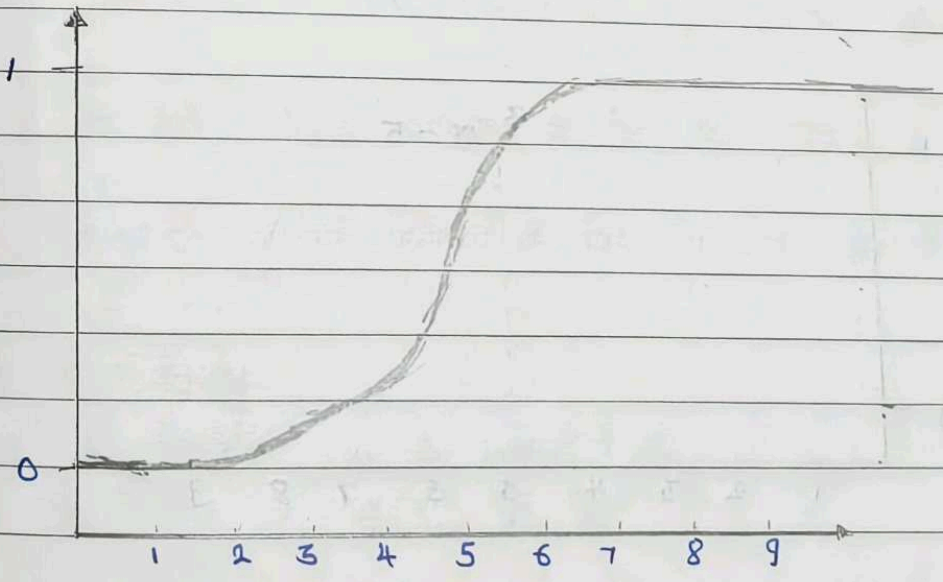
- a = slope (steepness of rise)
- c = center (inflection point) where curve transition.

Formula :

$$\mu(x) = \frac{1}{1 + e^{-a(x-c)}}$$

Usage : Models "slow to fast" or "small to large" transitions (like fuzzy "tall")

Example : Sigmoid ($a=2$, $C=5$) → gradually rises around $x=5$.

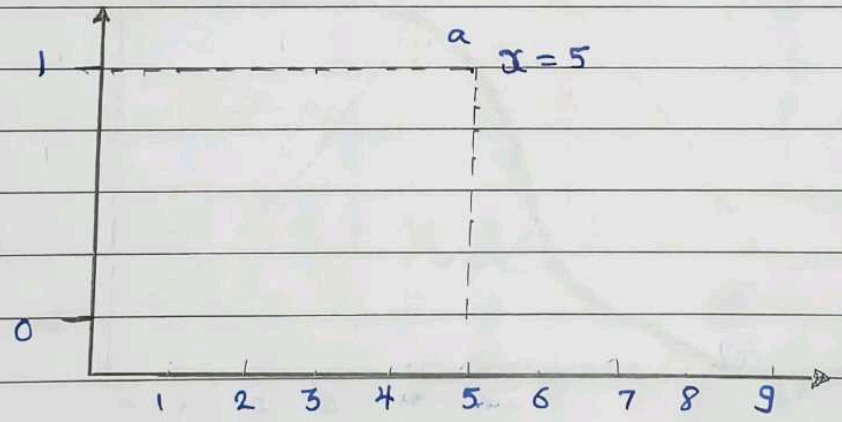


5. Singleton Membership Function

- Shape : Just one spike (like Dirac delta)
- Parameters :
  - a : The point where membership is 1
- Formula :

$$\mu(x) = \begin{cases} 1 & x = a \\ 0 & x \neq a \end{cases}$$

- Usage : Used in fuzzy databases & defuzzification when a crisp value is needed.
- Example: Singleton (5) → $\mu(x) = 1$ at $x = 5$, 0 everywhere else.



Assignment No : 4

Solve some mathematical data based problem using the steps of PSO

Particle Swarm Optimization (PSO) is a population-based optimization algorithm inspired by the social behaviour of birds flocking or fish schooling. In PSO, each potential solution is represented as a particle that moves through the search space by adjusting its position & velocity based on its own experience & that of neighboring particles. Each particle keeps track of its best position found so far (personal best) & is also influenced by the best position found by the entire swarm (global best). Through iterative updates, the swarm converges towards the optimal or near-optimal solution. PSO is widely used due to its simplicity, efficiency, & ability to handle complex, non-linear, & multidimensional optimization problems.

pbest : The best solution achieved so far by particle i.

gbest : The best solution achieved so far by any particle in the swarm

Position & Velocity

Position of Particle (i) is adjusted as

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$$

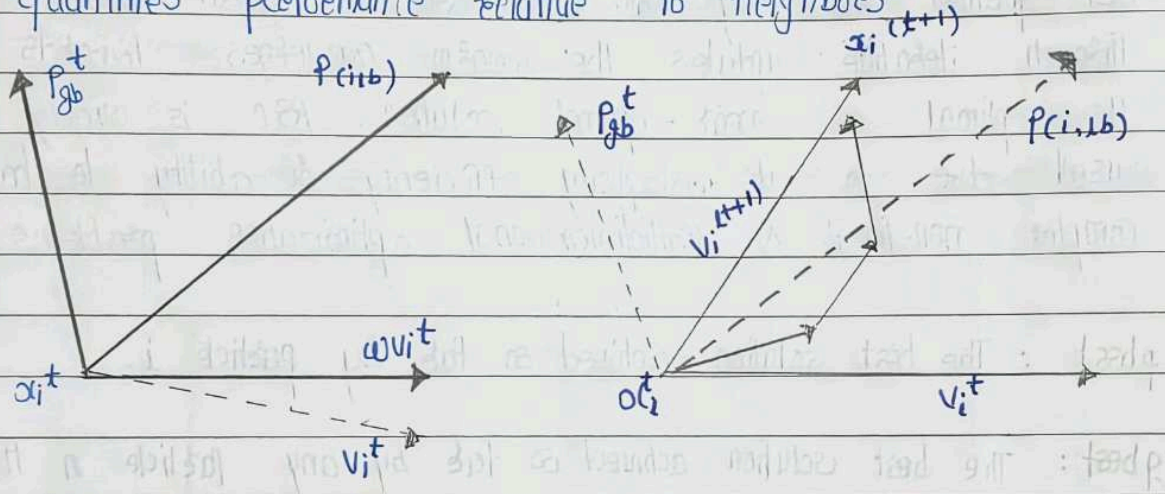
Velocity of Particle (i) is updated as follows.

$$v_i^{(t+1)} = \omega v_i^{(t)} + c_1 r_1 (P^t_{(i,b)} - x_i^t) + c_2 r_2 (P^t_{gb} - x_i^t)$$

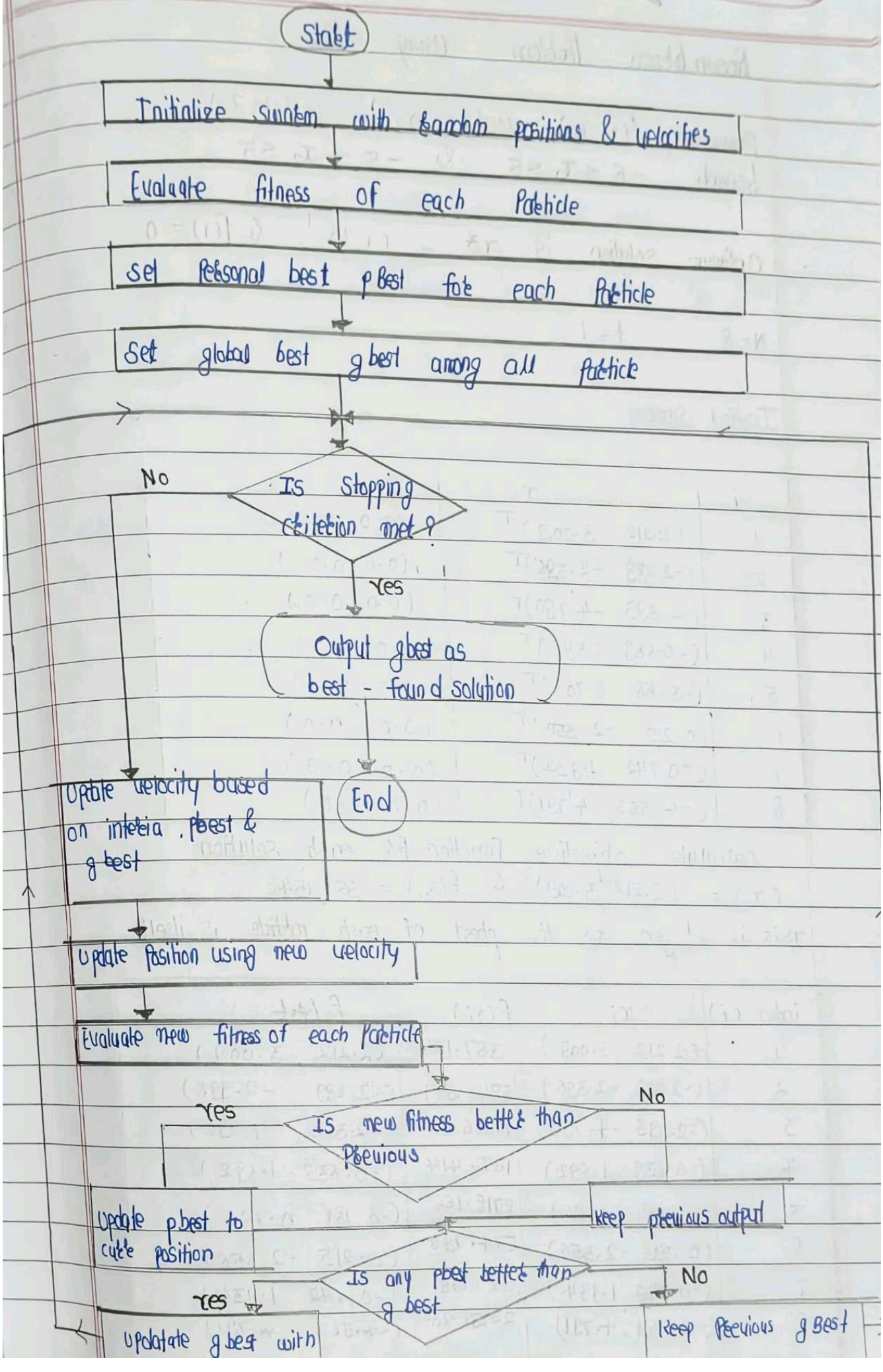
i is i^{th} particle

t is generation counter

- Momentum Part $\omega v_i^{(t)}$
  - inertia component
  - memory of previous flight direction
  - prevents particle from drastically changing direction.
- Cognitive Part $(P^t_{(i,b)} - x_i^{(t)})$
  - quantifies performance relative to past performance
  - memory of previous best position
  - Nostalgia.
- Social Part $c_2 r_2$ $(P^t_{gb} - x_i^t)$
  - quantifies performance relative to neighbors



Flowchart


Start
Initialize swarm with random positions & velocities
Evaluate fitness of each Particle
set Personal best pBest for each Particle
Set global best gbest among all Particle
Is Stopping Criterion met ?
No
Yes
Output gbest as best - found solution
End
Update velocity based on inertia, pbest & gbest
Update position using new velocity
Evaluate new fitness of each Particle
Is new fitness better than Previous
Yes
No
Update pbest to cute position
keep previous output
Is any pbest better than gbest
Yes
No
Update gbest with
keep Previous gBest

Rosenbrock Problem using PSO

Minimize $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

bounds $-5 \le x_1 \le 5$ & $-5 \le x_2 \le 5$

Optimum solution is $x^* = (1,1)^T$ & $f(x) = 0$

$N = 8$, $t = 1$

Initial swarm

i	x_i	v_i
1	$(2.212, 3.009)^T$	(0.0, 0.0)
2	$(-2.289, -2.396)^T$	(0.0, 0.0)
3	$(-2.393, -4.790)^T$	(0.0, 0.0)
4	$(-0.639, 1.692)^T$	(0.0, 0.0)
5	$(-3.168, 0.706)^T$	(0.0, 0.0)
6	$(0.215, -2.350)^T$	(0.0, 0.0)
7	$(-0.742, 1.934)^T$	(0.0, 0.0)
8	$(-4.563, 4.791)^T$	(0.0, 0.0)

calculate objective function for each solution

$(x_1) = (2.212, 3.009)^T$ & $f(x_1) = 357.154$

This is 1st gen so the pbest of each particle is itself

index (i)	x_i	$f(x_i)$	P best
1	(2.212, 3.009)	357.154	(2.212, 3.009)
2	(-2.289, -2.396)	5843.569	(-2.289, -2.396)
3	(-2.393, -4.790)	11066.800	(-2.393, -4.790)
4	(-0.639, 1.692)	167.414	(-0.639, 1.692)
5	(-3.168, 0.706)	8718.166	(-3.168, 0.706)
6	(0.215, -2.356)	574.796	(0.215, -2.356)
7	(-0.742, 1.934)	196.618	(-0.742, 1.934)
8	(-4.563, 4.791)	25731.235	(-4.563, 4.791)

The global best of the swarm is

P_{gb} = 4 (-0.639, 1.692)

The velocity of each particle is updated using

$$V_i^{(t+1)} = w V_i^{(t)} + c_1 r_1 (P^t_{(ib)} - x_i^t) + c_2 r_2 (P_{gb}^t - x_i^t)$$

Assume $w = 0.75$, $c_1 = 1.5$ & $c_2 = 2.0$

$x^{(1)}$ = (2.212, 3.009)

$V_i^{(1)}$ = (0.0, 0.0)

Pbest (1) = (2.212, 3.009)

$P_{gb}^{(1)}$ = (-0.639, 1.692), $r_1 = 0.661$ & $r_2 = 0.312$

$V_{1,1}^{(2)}$ = 0.75 x 0 + 1.5 x 0.661 (2.212 - 2.212) + 2.0 x 0.312 (-0.639 - 2.212) = -1.779

$V_{(1,2)}^{(2)}$ = 0.7 x 0 + 1.5 x 0.661 (3.009 - 3.009) + 2.0 x 0.312 (1.692 - 3.009) = 0.822

Updated velocity of each particle

Position Update

$$x_i^{(t+1)} = x_i^{(t)} + V_i^{(t+1)}$$

i	$x_i^{(2)}$	$V_i^{(2)}$	$f(x_i^2)$
1	(0.433, 2.187)	(-1.779, -0.822)	399.984
2	(-1.396, -0.184)	(0.893, 2.212)	460.648
3	(0.498, 5.893)	(2.890, 10.683)	2258.514
4	(-0.639, 1.692)	(0.000, 0.000)	167.417
5	(0.157, 1.879)	(3.010, 1.174)	345.35
6	(-0.779, 2.354)	(-0.994, 4.704)	308.580
7	(-0.590, 1.577)	(0.151, -0.357)	153.484
8	(2.928, -1.125)	(7.491, -5.916)	9406.994

Psedu Code.

Initialize a swarm of particles with random position & velocity

For each Particle:
    Evaluate fitness
    set pbest to its current position

set global best to best pBest among all particles

Repeat until stopping criterion is met
    For each particle:
        Update Velocity:
        $V[i] = \omega * V[i] + c_1 * r_1 * (pBest[i] - pos[i]) + c_2 * r_2 * (gbest - pos[i])$

        Update Position:
        $pos[i] = pos[i] + V[i]$

    Evaluate the new fitness of the particle.

    If current fitness < fitness (pBest [i]):
        Update pBest[i] = current position.

    If fitness (pBest [i] < fitness (gBest):
        Update gBest = pBest [i]

Output gBest as the best-found solution

Assignment No : 5

Write a program to solve a single objective function problem using Grey wolf optimization technique.

The Grey Wolf Optimization (GWO) algorithm is a nature-inspired metaheuristic technique based on the social hierarchy & hunting behavior of grey wolves in the wild. Proposed by Mirjalili in 2014, it mimics how grey wolves encircle, chase & attack their prey. The algorithm classifies wolves into four levels - alpha (leaders), beta, delta and omega representing different roles in decision-making & guidance. During optimization, the top three wolves guide the rest towards the prey (optimal solution) by updating positions based on these leaders. GWO is widely used for solving complex optimization problems due to its simplicity, balance between exploration & exploitation, & strong convergence ability.

Update Position

$$X(t) = \frac{X_1 + X_2 + X_3}{3}$$

best

$A_1 = 2a \text{ rand} - a$

$C_1 = 2 \text{ rand}$

$D_\alpha = |C_1 X_\alpha - X(t)|$

$X_1 = X_\alpha - A_1 D_\alpha$

and best

$A_2 = 2a \text{ rand} - a$

$C_2 = 2 \text{ rand}$

$D_\beta = |C_2 X_\beta - X(t)|$

$X_2 = X_\beta - A_2 D_{\beta_2}$

3rd best

$A_3 = 2at - a$

$C_3 = 2r$

$D_3 = |C_3 X_3 - X_t|$

$X_3 = X_3 - A_3 D_3$

"r/rand" is the random number between 0 & 1

$$a = 2\left(1 - \frac{\text{iteration}}{\text{max iteration}}\right)$$

Working steps of the GWO

1. Initialize the parameters : Population size, maximum iteration.
2. Find the best (X_α) 2nd best (X_β) & 3rd best (X_δ) positions.
3. For iteration = 1

- Compute $a = 2\left(1 - \frac{\text{iteration}}{\text{maxiteration}}\right)$
- Compute X_1, X_2, X_3
- Compute $X_{new} = (X_1 + X_2 + X_3)/3$
- check bounds
- Perform Greedy selection i.e $f(X_{new})$ is better or not. If yes update solution.

* Pseudo Code

Initialize the grey wolf population X_i $(i = 1, 2, 3, \ldots n)$

Initialize a, A, C.

Calculate the fitness of each search agent

X_α - the best search agent
X_β - the 2nd best search agent.
X_δ - the 3rd best search agent.

while (t < max number of iteration)
    for each search ager
        update the position of the current search ag
    end for

    Update a, A & C
    Calculate the fitness of all search agents
    Update X_α, X_β, X_δ
    t = t+1
end while

Minimize $f(x) = x_1^2 - x_1 x_2 + x_2^2 + 2x_1 + 4x_2 + 3$

where $-5 \le x_1, x_2 \le 5$

let population size = 5

No of cycle Iteration = 200

$x = L + \text{rand} * (U - L)$

x_1	x_2	$f(x_1, x_2)$
-4.5861	3.9930	65.0885
-0.0215	4.1747	37.1744
-0.8079	4.3295	41.5973
1.7389	2.6541	22.5470
.005	-2.6027	4.9692

$X_\alpha = [1.0005, -2.6027]$

$X_\beta = [1.7389, 2.6541]$

$X_\delta = [-0.0215, 4.1747]$

Iteration 1

$$a = 2\left(1 - \frac{\text{iteration}}{\text{max iteration}}\right)$$

$$= 2\left(1 - \frac{1}{200}\right)$$

$$= 1.99$$

For 1st wolf

$X_\alpha = X_\alpha - AD_\alpha$

$A = 2at - a$

$C = 2t$
$= 2(0.65)$

$D_\alpha = |1.30(1.005, -2.6027) - (-4.5861, 3.9930)|$
$= (5.88676, 7.3765)$

$X_\alpha = (1.005, -2.6027) - 0.1592(5.8865, 7.3765)$
$= (0.06333, -3.77764)$

Similarly

$X_\beta = (6.32036, 2.66161)$
$X_\delta = (-4.38818, 3.01221)$

$X_{new} = (X_\alpha + X_\beta + X_\delta)/3$
$= (0.66517, 0.65226)$

$f(x) = 65$
$f(x_{new}) = 7.3734$

If $f(x_{new}) < f(x)$
Accept X_{new}

$X_{1\,new}$	X_{2new}
0.66517	0.65226
2.39678	3.07265
0.48362	-0.26494
1.09022	-1.05407
0.41150	-0.91718

Assignment No : 6

* Write a program to ~~solve~~ simulate the working of Ant Colony Optimization.

Ant Colony Optimization (ACO) is a nature inspired algorithm based on the foraging behavior of ants. In nature ants find the shortest path between their nest & a food source by depositing pheromones on the ground. Over time, shorter paths accumulate stronger pheromone trails, attracting more ants & reinforcing the optimal route. Similarly in ACO, artificial "ants" explore possible solutions to an optimization problem, updating pheromone values to guide future searches. This iterative process helps efficiently find near-optimal solutions to complex problems like routing, scheduling, & network optimization.

* Steps of ACO

1) Initialize Parameters - Set the number of ants, pheromone evaporation rate & other constants

2. Initialize Pheromone Trails - Assign an initial pheromone value to all possible paths.

3. Construct Solutions - Each ant builds a solution by moving from one node to another based on pheromone intensity & heuristic information (like distance)

4. Evaluate Solutions - Calculate the quality (fitness) of each ants solution.

Assignment No : 8

* Write a program to ~~solve~~ simulate the working of Ant Colony Optimization.

Ant Colony Optimization (ACO) is a nature inspired algorithm based on the foraging behaviour of ants. In nature ants find the shortest path between their nest & a food source by depositing pheromones on the ground. Over time, shorter paths accumulate stronger pheromone trails, attracting more ants & reinforcing the optimal route. Similarly, in ACO, artificial "ants" explore possible solutions to an optimization problem, updating pheromone values to guide future searches. This iterative process helps efficiently find near-optimal solutions to complex problems like routing, scheduling, & network optimization.

* Steps of ACO

1) Initialize Parameters - Set the number of ants, pheromone evaporation rate & other constants.

2. Initialize Pheromone Trails - Assign an initial pheromone value to all possible paths.

3. Construct Solutions - Each ant build a solution by moving from one node to another based on pheromone intensity & heuristic information (like distance)

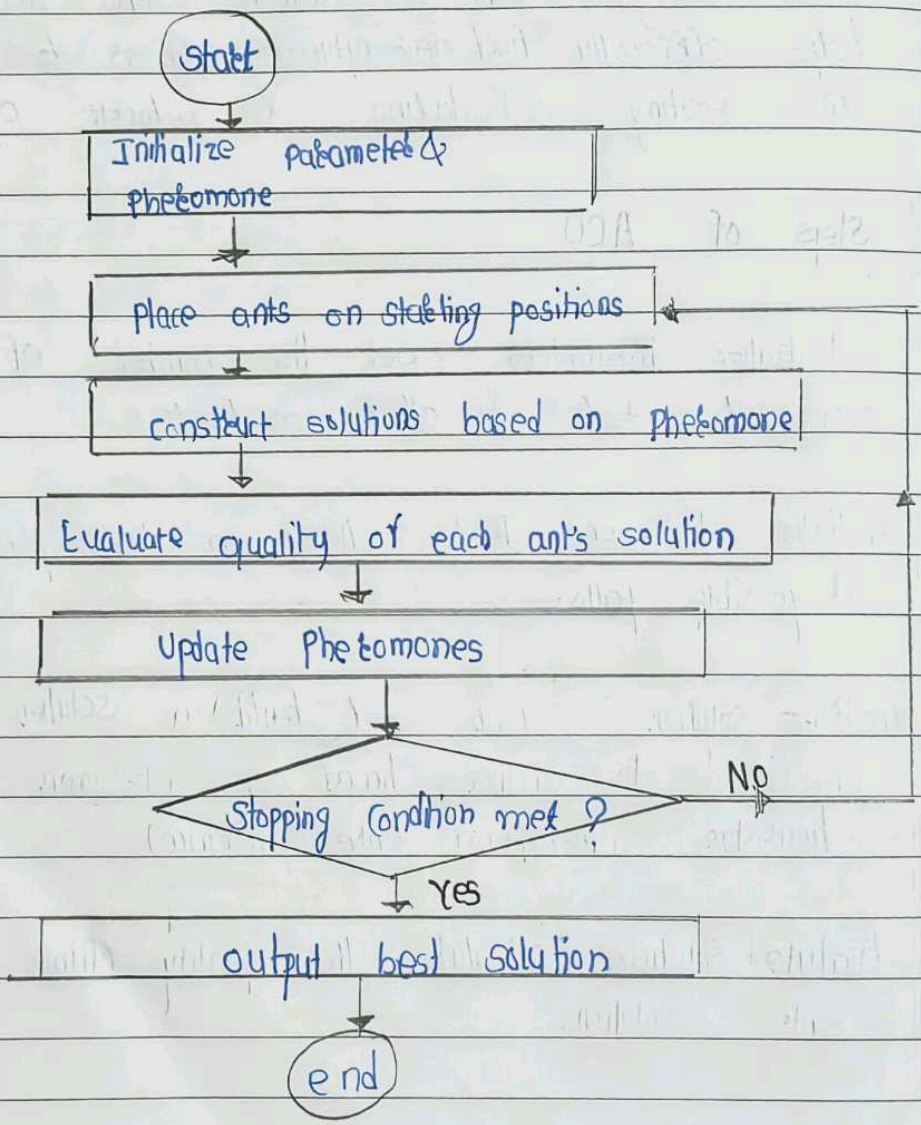
4. Evaluate Solutions - Calculate the quality (fitness) of each ants solution.

5. Update Pheromones -

- Evaporate some pheromone on all paths to avoid stagnation
- Deposit pheromone on paths used by better-performing ants to reinforce good solutions.

6. Check stopping Condition - If the maximum number of iterations or convergence is reached, stop otherwise go back to step 3.

7. Output the best solution - Report the path or solution with the highest pheromone level or best performance.



Example :

	1	2	3	4
1	0	5	15	4
2	5	0	4	8
3	15	4	0	1
4	4	8	1	0

cost matrix

	1	2	3	4
1	0	4	10	3
2	4	0	1	2
3	10	1	6	1
4	3	2	1	0

Pheromone matrix

$$\Delta\tau^k_{i,j} = \begin{cases} \frac{1}{L^k} & k^{th} \text{ ant leaves on the edge } i,j \\ 0 & \text{otherwise} \end{cases}$$

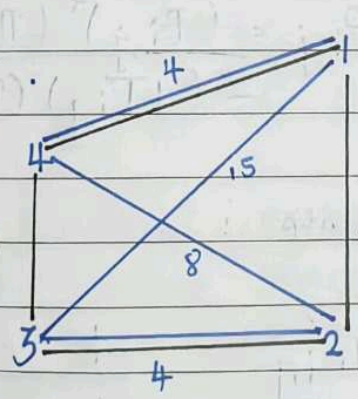
L is length of Path

$$\tau^k_{i,j} = \sum_{k=1}^{m} \Delta\tau^k_{i,j} \quad \text{without vapourization}$$

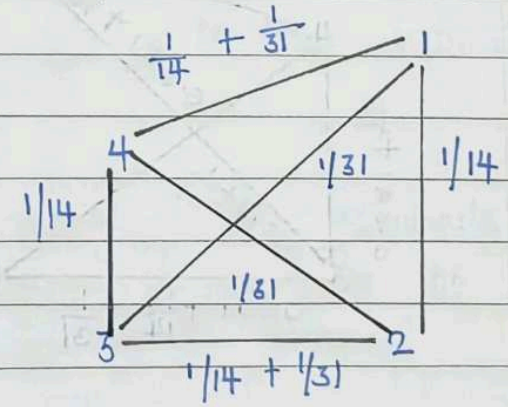
$$\tau^k_{i,j} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau^k_{i,j} \quad \text{... with vapourization}$$

where ρ is evaporation rate

Consider two ants blue & black



cost Graph



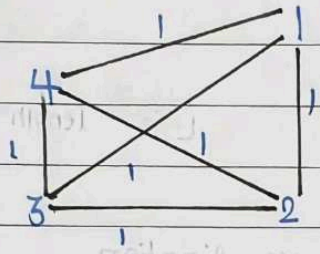
Pheromone Graph

$L_1 = 14 \rightarrow \Delta\tau^1_{ij} = \frac{1}{14}$

$L_2 = 31 \rightarrow \Delta\tau^2_{ij} = \frac{1}{31}$

$$\tau_{i,j} = \sum_{k=1}^{m} \Delta\tau^k_{i,j}$$

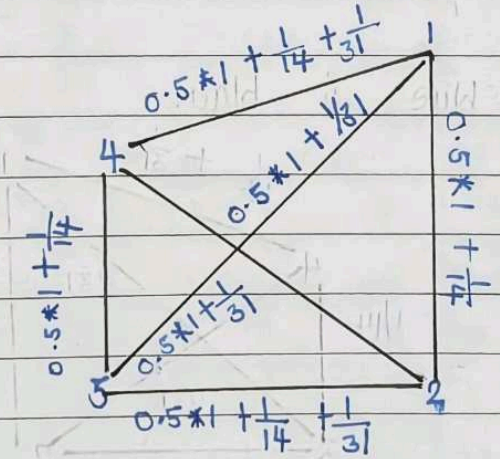
Consider Initial Pheromone for Evaporization



$$\tau_{i,j} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau^k_{i,j}$$

Pheromone Graph

Pheromone Graph Becomes. For $\rho = 0.5$



Calculating Probability

$$P_{i,j} = \frac{(\tau_{i,j})^{\alpha}\,(\eta_{i,j})^{\beta}}{\sum\left((\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}\right)}$$

where :

$$\eta_{i,j} = \frac{1}{L_{ij}}$$

Assignment No : 7

Implement Selection algorithms

Selection is the process of creating the population for next generation from the current generation

Canonical Selection :

In this techniques, fitness is defined for the i^{th} individual as follows

$$\text{fitness}(i) = \frac{f_i}{\bar{F}}$$

where f_i is the evaluation associated with the i^{th} individual in the population.

$\bar{F}$ is the average evaluation of all individuals in the population size N and is defined as follows

$$\bar{F} = \frac{\sum_{i=1}^{N} f_i}{N}$$

In an iteration, we calculate $\frac{f_i}{\bar{F}}$ for all individuals in the current population.

In Canonical selection, the probability that individuals in the current population are copied & placed in the mating pool is Proportional to their fitness

We select N_p number of values where N_p is f% of N

$N_p \ll N$

* Steps / Flow

Step 1 : Calculate $\bar{F}$

Step 2 : Calculate fitness value for each f_i

Step 3 : Select N_p best values

Example :

f_i	1.01	2.11	3.11	4.01	1.91	1.93

$$\bar{F} = \frac{\sum_{i=1}^{N} f_i}{N} = \frac{1.01 + 2.11 + 3.11 + 4.01 + 1.91 + 1.93}{6}$$

$$= \frac{14.08}{6}$$

$$= 2.34$$

$$\text{Fitness}(i) = \frac{f_i}{\bar{F}}$$

$$\text{Fitness}(0) = \frac{f_0}{\bar{F}} = \frac{1.01}{2.34} = 0.43$$

Similarly calculate fitness for all points & select first N_p best / greatest points.

Roulette wheel Selection

Greater the fitness value greater is the probability to get selected for Breeding.

Mechanism:

Top surface area of wheel is divided into N parts in proportion to the fitness values $F_1, F_2, \ldots F_N$

The wheel is rotated in a particular direction & a fixed pointer is used to indicate the winning area when it stops rotation.

A particular sub-area representing a GA - solution is selected to be winner probabilistically & the probability that i^{th} area will be declared

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

* Example.

Individual	Fitness value	p_i
1	1.01	0.05
2	2.11	0.09
3	3.11	0.13
4	4.01	0.17
5	4.66	0.20
6	1.91	0.08
7	1.93	0.08
8	4.51	0.20

Implementation

Input : A population of size N with their fitness values.

Output : A mating pool of size Np

Steps :

1) Compute $p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$. $\forall_i = 1, 2, \ldots N$

2) Calculate the cumulative probability for each of the individual starting from the top of the list, that is

$$P_i = \sum_{j=1}^{i} p_j \quad \text{for all } j = 1, 2 \ldots N$$

3) Generate a random number say r between 0 & 1

4) Select the j^{th} individual such that $P_{j-1} < r \leq P_j$

5) Repeat step 3-4 to select Np individuals

6) End.

Individual	f_i	P_i	ξ	T
1	0.05	0.05	0.26	1
2	0.09	0.14	0.04	1
3	0.13	0.27	0.48	11
4	0.17	0.44	0.43	1
5	0.20	0.64	0.09	11
6	0.08	0.72	0.30	
7	0.08	0.80	0.61	
8	0.20	1.0	0.89	1

Problem with Roulette-wheel selection

- The individual with higher fitness values will guard the other to be selected for mating
- This leads to a lesser diversity & hence fewer scope towards exploring the alternative solution & also premature convergence or early convergence with local optimal solution.

3) Rank - based Selection

The process in rank selection consist of two steps.

1) Individuals are arranged in an ascending order of their fitness values. The individual which has the lowest value of fitness is assigned rank 1, & other individuals are ranked accordingly.

2) The proportionate based selection scheme is then followed based on the assigned rank.

NOTE :

The % area to be occupied by a particular individual i, is given by

$$\frac{r_i}{\sum_{i=1}^{N} r_i}$$

where r_i indicates the ranks of the i^{th} individual.

- Two or more individuals with the same fitness values should have same rank.

Example

- lets take population of four individuals with the fitness values.

$f_1 = 0.40$, $f_2 = 0.05$, $f_3 = 0.03$ & $f_4 = 0.02$

- Their proportionate area on the wheel according to RW selection are 80%, 10%, 6% & 4%.

- Their ranks are as follows

Individual	Fitness	RW (Area)	Rank	RS (Area)
1	0.4	80%	4	40%
2	0.05	10%	3	30%
3	0.03	6%	2	20%
4	0.02	4%	1	10%

$$\text{RS Area} = \frac{f_i}{\sum_{1}^{N} f_i}$$

$$\% \text{ Area of (1)} = \frac{f_1}{\sum_{1}^{N} f_i} = \frac{4}{4+3+2+1} = \frac{4}{10} = 0.4$$

Implementation.

Input : A population of size N with their fitness values

Output : A mating pool of size Np

Steps :

1) Arrange all individuals in ascending order of their fitness value

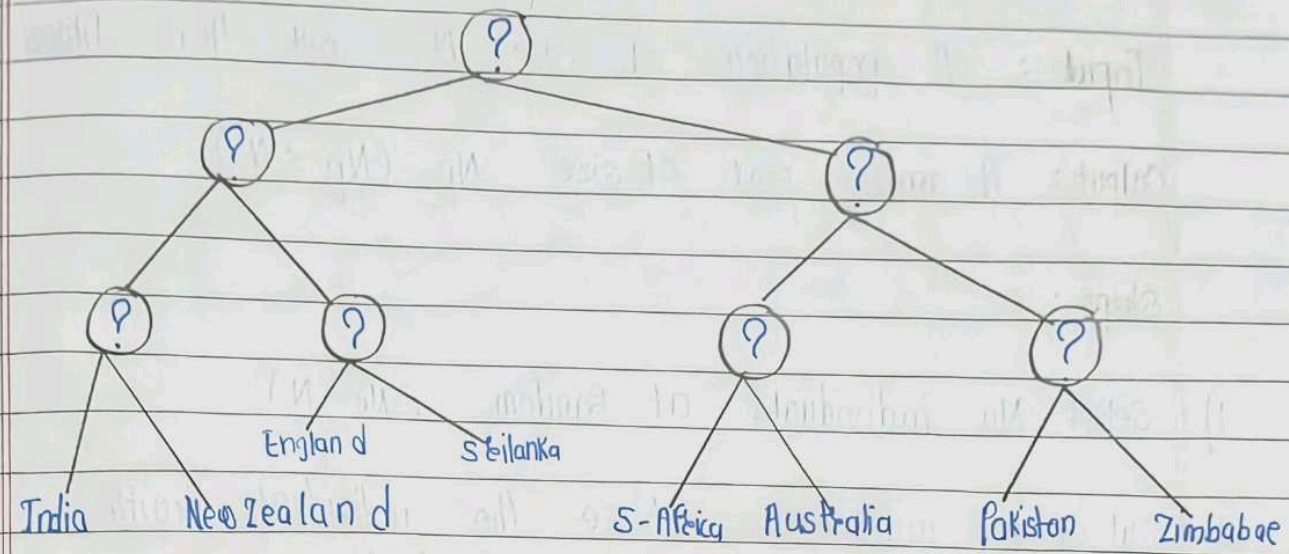
2) Rank the individuals according to their position in the order, that is, the worst will have rank 1, the next rank 2 & best will have rank N.

3) Apply Roulette-wheel selection but based on their assigned ranks. For example the probability p_i of the ith individual would be

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

4) Stop

4) Tournament Selection.



1) In this scheme, we select the tournament size n at random.

2) We pick n individuals from the population, at random determine the best one in terms of their fitness values.

3) The best individual is copied into the mating pool

4) Thus, in this scheme only one individual is selected per tournament & Np tournaments are to be played to make the size of mating pool equals to Np.

Implementation

Input : A population of size N with their fitness value

Output : A mating pool of size Np ($Np \leq N$)

Steps :

1) Select Nu individuals at random ($Nu < N$)

2) Out of Nu individuals, choose the individual with highest fitness value as the winner

3) Add the winner to the mating pool, which is initially empty.

4) Repeat steps 1-3 until the mating pool contains Np individuals.

5) Stop.

Example.

N = 8 , Nu = 2 , Np = 8

Individual	1	2	3	4	5	6	7	8
Fitness	1.0	2.1	3.1	4.0	4.6	1.9	1.8	4.5

Output.

Trial	Individuals	Selected
1	2, 4	4
2	3, 8	8
3	1, 3	3
4	4, 5	5
5	1, 6	6
6	1, 2	2
7	4, 2	4
8	8, 3	8

* Steady state selection

Steps:

1. Nu individuals are selected at random
2. Nu individuals with the worst fitness values are replaced with Nu individuals selected in step 1 & are added into the mating pool.

This completes the selection procedure for one iteration. Repeat the iteration until the mating pool of desired size is obtained.

Assignment No : 8

X Implement Different Crossover Techniques in Binary coded GA.

1) Single Point Crossover.

- Here we select the K-point lying between 1 & L Let it be K
- A single crossover point at K on both parents strings is selected
- All data beyond that point in either string is swapped between the two parents.
- The Resulting strings are the chromosomes of the offspring produced.

Example.

Parent 1 : | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

Parent 2 : | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

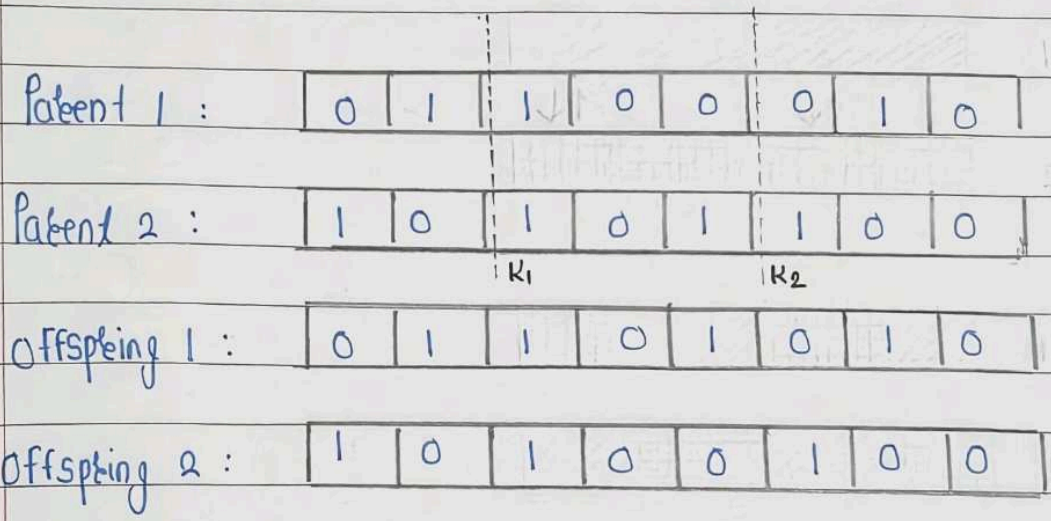
cross-over point K

Offspring 1 : | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

Offspring 2 : | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

2) Two-Point Crossover

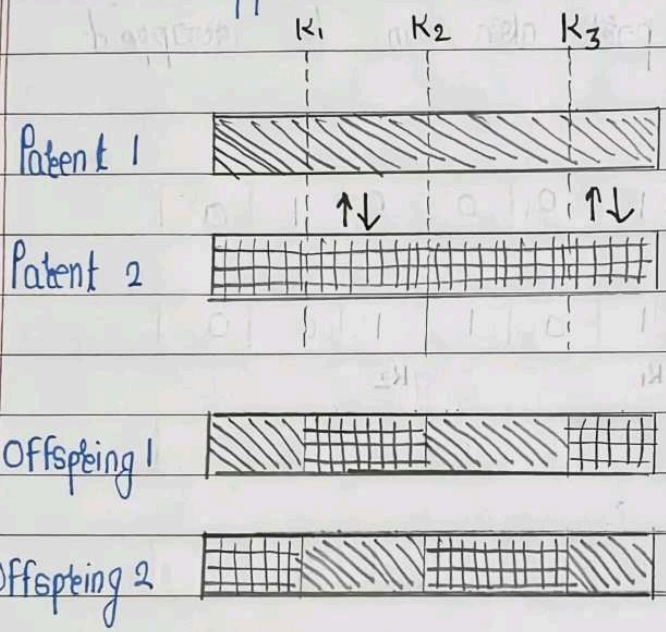
- In this scheme, we select two different crossover points K_1 & K_2 lying between 1 & L at random such that $K_1 \neq K_2$
- The middle parts are swapped between the two strings.
- Alternatively, left & right parts also can be swapped



3) Multipoint Crossover

1) In case of multipoint crossover, a number of crossover points are selected along the length of the string. at random.

2) the bits lying between alternate pairs of sites are then swapped



4) Uniform Crossover.

- Uniform Crossover is a more general version of the multi-point crossover.
- In this scheme, at each bit position of the parent string, we toss a coin (with a certain probability P_s) to determine whether there will be swap of the bits or not.
- The two bits are then swapped or remain unaltered accordingly.

Parent 1	1	1	0	0	0	1	0	1	1
Parent 2	0	1	1	0	0	1	1	1	0

Coin Tossing: (1) (0) (0) (1) (1) (1) (0) (1) (1)

Offspring 1	1	1	1	0	0	1	1	1	1
Offspring 2	0	1	0	0	0	0	0	1	0

5) Uniform Crossover with Crossover mask.

- Here each gene is created in the offspring by copying the corresponding gene from one of the other parent chosen according to a random generated binary crossover mask of the same length as the chromosome.
- where there is a 1 in the mask, the gene is copied from the first parent.
- when there is a 0 in the mask, the gene is copied from the second parent.
- The reverse is followed to create another offsprings