Lab - 2

Particle Swarm Optimization

Algorithm

Initialize

Swarm size (N), problem dimensionality (D), max iterations

Assign random positions and velocities for each particle

Step 1 :- Randomly initialize, swarm population of N particles $X_i (i = 1, 2, \ldots, n)$

Step 2 :- Select hyper parameter, values w, c1 and c2

Step 3 :- For iter in range (max_iter):

for i in range (N):

(a) Compute new velocity of ith particle

swarm[i].velocity = N * swarm[i].velocity + r1 * c1 * swarm[i].bestPos - swarm[i].position) + r2 * c2 * (best_pos_swarm - swarm[i].position)

(b) compute new position of ith particle using its new velocity

swarm[i].position += swarm[i].velocity

(c) if position is not in range [minx, maxx] then clip it

if swarm[i].position < minx :

swarm[i].position = minx

elif swarm[i].position > maxx :

swarm[i].position = maxx

(d) Update new best of this particle and new of swarm

if swa Incentive to coding of design

variables, sm[i].fitness < swarm[i]
best_fitness
swarm[i].bestFitness = swarm[i].fitness
swarm[i].bestPos = swarm[i].position
end for
end for

Step 4 :- Return best particle of swarm

Program

```
import numpy as np
def objective_function(x):
    return np.sum(x**2)

class Particle:
    def __init__(self, dimensions, minx, maxx):
        self.position = np.random.uniform(minx, maxx, dimensions)
        self.velocity = np.random.uniform(minx, maxx, dimensions)
        self.best_position = np.copy(self.position)
        self.best_fitness = objective_function(self.position)
        self.Fitness = self.bestFitness

def PSO(n_particles, dimensions, max_iter, minx, maxx, w, c1, c2):
    swarm = [Particle(dimensions, minx, maxx)
             for _ in range(n_particles)]
    best_fitness_swarm = float('inf')
    best_pos_swarm = None
    for iter in range(max_iter):
        for particle in swarm:
            r1, r2 = np.random.rand(dimensions),
                     np.random.rand(dimensions)
```

```
particle.velocity = (w * particle_velocity +
                    (r1 * c1 * (particle.best_
                    position - particle.position)
                    c2 * r2 * (best_program - particle.positio
                    if best_program is not None else 0))
particle.position += particle.velocity
particle.position = np.clip(particle.position, minx, ma
particle.fitness = objective_function(particle.positio
if particle.fitness < best_fitness_swarm:
    best_fitness_swarm = particle.fitness
    best_pos_swarm = np.copy(particle.position)
return best_pos_swarm, best_fitness_swarm
```

```
n_particles = 30
dimensions = 2
maxiter = 100
minx, maxx = -10, 10
w = 0.5
c1 = 1.5
c2 = 1.5
best_position, best_fitness = pso(n_particles, dimensions,
        maxiter, minx, maxx, w, c1, c2)
print("Best position found:", best_position)
print("Best fitness found:", best_fitness)
```

Output :-

Best fitness found: [-1.288e-12 -8.18e-13]

Best fitness found: 2.33e-24