1BM23CS212, Nidhi Pramod Nambiar

Application Problem

Optimize a system that has multiple conflicting objectives, such as minimizing cost and maximizing performance, in a design task (e.g., Manufacturing design, architecture)

Source code

```
import numpy as np
import matplotlib.pyplot as plt

# --------------------------
# 1. Define the problem
# --------------------------
# We will maximize a performance function while penalizing high cost.

def performance_function(position):
    """Performance to maximize"""
    x, y = position
    return np.sin(x) * np.cos(y)  # Peaks and valleys to test optimizer

def cost_function(position):
    """Cost to minimize (used as a penalty)"""
    x, y = position
    return x**2 + y**2  # Higher away from origin

def objective_function(position, alpha=0.1):
    """
    Combined function to MAXIMIZE.
    alpha controls how strongly cost is penalized.
    """
    perf = performance_function(position)
```

```
    cost = cost_function(position)
    fitness = perf - alpha * cost  # maximize performance, penalize cost
    return fitness

# --------------------------
# 2. Initialize PSO parameters
# --------------------------
num_particles = 30
num_dimensions = 2
num_iterations = 100

w = 0.7      # inertia weight
c1 = 1.5     # cognitive coefficient
c2 = 1.5     # social coefficient

bounds = [-5, 5]  # search space limits

# --------------------------
# 3. Initialize particles
# --------------------------
positions = np.random.uniform(bounds[0], bounds[1], (num_particles, num_dimensions))
velocities = np.random.uniform(-1, 1, (num_particles, num_dimensions))

# Each particle's personal best (start with its initial position)
personal_best_positions = np.copy(positions)
personal_best_scores = np.array([objective_function(p) for p in positions])

# Initialize the global best (highest fitness)
global_best_position = personal_best_positions[np.argmax(personal_best_scores)]
global_best_score = np.max(personal_best_scores)
```

```
# --------------------------
# 4–6. Main PSO loop
# --------------------------
convergence_curve = []

for iteration in range(num_iterations):
    for i in range(num_particles):
        # Evaluate fitness (we are maximizing)
        fitness = objective_function(positions[i])

        # Update personal best (if better)
        if fitness > personal_best_scores[i]:
            personal_best_scores[i] = fitness
            personal_best_positions[i] = positions[i]

        # Update global best
        if fitness > global_best_score:
            global_best_score = fitness
            global_best_position = positions[i]

    # Update velocity and position
    r1, r2 = np.random.rand(num_particles, num_dimensions),
np.random.rand(num_particles, num_dimensions)
    cognitive = c1 * r1 * (personal_best_positions - positions)
    social = c2 * r2 * (global_best_position - positions)
    velocities = w * velocities + cognitive + social

    # Update positions
    positions += velocities

    # Apply boundary constraints
```

```
    positions = np.clip(positions, bounds[0], bounds[1])

    # Track global best for convergence
    convergence_curve.append(global_best_score)

    # Optional: print progress
    if iteration % 10 == 0:
        print(f"Iteration {iteration}/{num_iterations}, Best Fitness: {global_best_score:.6f}")

# --------------------------
# 7. Output results
# --------------------------
print("\nOptimization Complete!")
print("Best Position:", global_best_position)
print("Best Fitness (Performance - Penalty):", global_best_score)
print("Performance:", performance_function(global_best_position))
print("Cost:", cost_function(global_best_position))
```

Output:

```
Iteration 0/100, Best Fitness: 0.608601
Iteration 10/100, Best Fitness: 0.789188
Iteration 20/100, Best Fitness: 0.792573
Iteration 30/100, Best Fitness: 0.793264
Iteration 40/100, Best Fitness: 0.794308
Iteration 50/100, Best Fitness: 0.794354
Iteration 60/100, Best Fitness: 0.794549
Iteration 70/100, Best Fitness: 0.794575
Iteration 80/100, Best Fitness: 0.794582
Iteration 90/100, Best Fitness: 0.794582

Optimization Complete!
Best Position: [1.29566279 0.00601934]
Best Fitness (Performance - Penalty): 0.7945822816857704
Performance: 0.9623714915238872
Cost: 1.6787782865072953
```