Multi-Robot Task Allocation using Particle Swarm optimization

```
import numpy as np


num_robots = 3     # number of robots
num_tasks = 5      # number of tasks


# Example cost matrix (rows = robots, cols = tasks)
cost = np.array([
    [9, 2, 7, 8, 6],
    [6, 4, 3, 7, 5],
    [5, 8, 1, 8, 7]
])


num_particles = 5     # number of candidate solutions
max_iter = 10         # number of iterations
w = 0.5               # inertia weight
c1 = 1.5              # personal learning factor
c2 = 1.5              # social learning factor


# ---------------------------
# Fitness Function
# ---------------------------
def fitness(position):
    """Calculate total cost for a given assignment."""
    assignment = np.round(position).astype(int)
    assignment = np.clip(assignment, 0, num_robots - 1)
    total_cost = sum(cost[assignment[j], j] for j in range(num_tasks))
    return total_cost
```

```
# ---------------------------
# Initialize Swarm
# ---------------------------
positions = np.random.uniform(0, num_robots - 1, (num_particles, num_tasks))
velocities = np.zeros_like(positions)

personal_best = positions.copy()
personal_best_score = np.array([fitness(p) for p in positions])

best_idx = np.argmin(personal_best_score)
global_best = personal_best[best_idx].copy()
global_best_score = personal_best_score[best_idx]

# ---------------------------
# Main PSO Loop
# ---------------------------
print("Initial Global Best Cost:", global_best_score)
print("="*60)

for t in range(max_iter):
    for i in range(num_particles):
        # Random coefficients
        r1, r2 = np.random.rand(num_tasks), np.random.rand(num_tasks)

        # Update velocity and position
        velocities[i] = (w * velocities[i] +
                         c1 * r1 * (personal_best[i] - positions[i]) +
                         c2 * r2 * (global_best - positions[i]))

        positions[i] += velocities[i]
        positions[i] = np.clip(positions[i], 0, num_robots - 1)
```

```
        # Evaluate fitness
        score = fitness(positions[i])

        # Update personal best
        if score < personal_best_score[i]:
            personal_best_score[i] = score
            personal_best[i] = positions[i].copy()

    # Update global best
    best_idx = np.argmin(personal_best_score)
    if personal_best_score[best_idx] < global_best_score:
        global_best_score = personal_best_score[best_idx]
        global_best = personal_best[best_idx].copy()

    # ---------------------------
    # Print iteration details
    # ---------------------------
    print(f"\nIteration {t+1}:")
    for i in range(num_particles):
        print(f"  Particle {i+1}: pbest = {np.round(personal_best[i]).astype(int)}, "
              f"pbest_cost = {personal_best_score[i]}")
    print(f"  --> Global Best (gbest) = {np.round(global_best).astype(int)}, "
          f"Cost = {global_best_score}")
    print("-"*60)

# ---------------------------
# Final Result
# ---------------------------
print("\nFINAL RESULTS")
print("Optimal Global Assignment (task -> robot):", np.round(global_best).astype(int))
```

```
print("Minimum Total Cost:", global_best_score)
```

```
Initial Global Best Cost: 24
============================================================

Iteration 1:
  Particle 1: pbest = [1 0 1 1 2], pbest_cost = 25
  Particle 2: pbest = [1 1 2 2 1], pbest_cost = 24
  Particle 3: pbest = [1 0 2 2 1], pbest_cost = 22
  Particle 4: pbest = [1 1 1 1 1], pbest_cost = 25
  Particle 5: pbest = [1 1 2 2 1], pbest_cost = 24
  --> Global Best (gbest) = [1 0 2 2 1], Cost = 22
------------------------------------------------------------

Iteration 2:
  Particle 1: pbest = [1 0 2 2 2], pbest_cost = 24
  Particle 2: pbest = [1 1 2 2 1], pbest_cost = 24
  Particle 3: pbest = [1 0 2 2 1], pbest_cost = 22
  Particle 4: pbest = [1 0 2 2 0], pbest_cost = 23
  Particle 5: pbest = [1 0 2 2 1], pbest_cost = 22
  --> Global Best (gbest) = [1 0 2 2 1], Cost = 22
------------------------------------------------------------

Iteration 3:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 1 2 1 1], pbest_cost = 23
  Particle 3: pbest = [1 0 2 2 1], pbest_cost = 22
  Particle 4: pbest = [1 0 2 2 0], pbest_cost = 23
  Particle 5: pbest = [1 0 2 2 1], pbest_cost = 22
  --> Global Best (gbest) = [1 0 2 1 1], Cost = 21
------------------------------------------------------------

Iteration 4:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 1 2 1 1], pbest_cost = 23
  Particle 3: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 4: pbest = [1 0 2 1 0], pbest_cost = 22
  Particle 5: pbest = [1 0 2 2 1], pbest_cost = 22
  --> Global Best (gbest) = [1 0 2 1 1], Cost = 21
------------------------------------------------------------

Iteration 5:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
```

```
-----------------------------------------------------------

Iteration 10:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 3: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 4: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 5: pbest = [1 0 2 1 1], pbest_cost = 21
  --> Global Best (gbest) = [1 0 2 1 1], Cost = 21
-----------------------------------------------------------

FINAL RESULTS
Optimal Global Assignment (task -> robot): [1 0 2 1 1]
Minimum Total Cost: 21
```

```
Iteration 5:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 3: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 4: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 5: pbest = [1 0 2 1 1], pbest_cost = 21
  --> Global Best (gbest) = [1 0 2 1 1], Cost = 21
-----------------------------------------------------------

Iteration 6:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 3: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 4: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 5: pbest = [1 0 2 1 1], pbest_cost = 21
  --> Global Best (gbest) = [1 0 2 1 1], Cost = 21
-----------------------------------------------------------

Iteration 7:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 3: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 4: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 5: pbest = [1 0 2 1 1], pbest_cost = 21
  --> Global Best (gbest) = [1 0 2 1 1], Cost = 21
-----------------------------------------------------------

Iteration 8:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 3: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 4: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 5: pbest = [1 0 2 1 1], pbest_cost = 21
  --> Global Best (gbest) = [1 0 2 1 1], Cost = 21
-----------------------------------------------------------

Iteration 9:
  Particle 1: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 2: pbest = [1 0 2 1 1], pbest_cost = 21
  Particle 3: pbest = [1 0 2 1 1], pbest_cost = 21
```