BIS LAB 5

17/10/2025

Sharada Koundinya (1BM23CS310)

GREY WOLF OPTIMIZATION:

CODE:

```python
import numpy as np

num_tasks = 20          # Number of tasks
num_vms = 5             # Number of virtual machines
num_wolves = 15         # Population size
max_iter = 50           # Maximum iterations

# Random task sizes
task_load = np.random.randint(1000, 10000, num_tasks)

# Random VM speeds
vm_speed = np.random.randint(500, 2000, num_vms)

# Fitness Function
def fitness(position):
    """
    position[i] = VM index assigned to task i
    Fitness = total makespan (time to finish all tasks)
    """
    loads = np.zeros(num_vms)
    for i, vm in enumerate(position.astype(int)):
        loads[vm] += task_load[i] / vm_speed[vm]
    return np.max(loads)

wolves = np.random.randint(0, num_vms, (num_wolves, num_tasks))
alpha, beta, delta = None, None, None
alpha_score, beta_score, delta_score = np.inf, np.inf, np.inf

for t in range(max_iter):
    a = 2 - 2 * t / max_iter

    for i in range(num_wolves):
        score = fitness(wolves[i])

        if score < alpha_score:
            alpha_score, alpha = score, wolves[i].copy()
        elif score < beta_score:
            beta_score, beta = score, wolves[i].copy()
        elif score < delta_score:
            delta_score, delta = score, wolves[i].copy()
    for i in range(num_wolves):
        for j in range(num_tasks):
            r1, r2 = np.random.rand(), np.random.rand()
            A1, C1 = 2*a*r1 - a, 2*r2
            D_alpha = abs(C1 * alpha[j] - wolves[i][j])
            X1 = alpha[j] - A1 * D_alpha

            r1, r2 = np.random.rand(), np.random.rand()
            A2, C2 = 2*a*r1 - a, 2*r2
            D_beta = abs(C2 * beta[j] - wolves[i][j])
            X2 = beta[j] - A2 * D_beta

            r1, r2 = np.random.rand(), np.random.rand()
            A3, C3 = 2*a*r1 - a, 2*r2
            D_delta = abs(C3 * delta[j] - wolves[i][j])
            X3 = delta[j] - A3 * D_delta

            new_pos = (X1 + X2 + X3) / 3
            wolves[i][j] = np.clip(round(new_pos), 0, num_vms - 1)

best_allocation = alpha.astype(int)
best_makespan = alpha_score

print("Best Makespan:", best_makespan)
print("Best Task Allocation (task → VM):")
for i, vm in enumerate(best_allocation):
    print(f"  Task {i+1} → VM {vm+1}")
```

OUTPUT:

```
Best Makespan: 21.60720720720721
Best Task Allocation (task → VM):
  Task 1 → VM 2
  Task 2 → VM 1
  Task 3 → VM 4
  Task 4 → VM 5
  Task 5 → VM 3
  Task 6 → VM 5
  Task 7 → VM 4
  Task 8 → VM 1
  Task 9 → VM 1
  Task 10 → VM 5
  Task 11 → VM 4
  Task 12 → VM 3
  Task 13 → VM 4
  Task 14 → VM 2
  Task 15 → VM 3
  Task 16 → VM 5
  Task 17 → VM 4
  Task 18 → VM 2
  Task 19 → VM 1
  Task 20 → VM 4
```