Lab-4

Cuckoo Search (CS)

```
import numpy as np

import math  # Import the standard math module

def levy_flight(Lambda):
    sigma = (math.gamma(1 + Lambda) * math.sin(math.pi *
Lambda / 2) /
             (math.gamma((1 + Lambda) / 2) * Lambda * 2
** ((Lambda - 1) / 2))) ** (1 / Lambda)
    u = np.random.normal(0, sigma, 1)
    v = np.random.normal(0, 1, 1)
    step = u / abs(v) ** (1 / Lambda)
    return step


def cuckoo_search(obj_function, bounds, n=25, pa=0.25,
max_iter=100):
    # Initialize nests
    dim = len(bounds)
    nests = np.random.rand(n, dim)
    for i in range(dim):
        nests[:, i] = nests[:, i] * (bounds[i][1] -
bounds[i][0]) + bounds[i][0]
    fitness = np.array([obj_function(nest) for nest in
nests])

    # Start optimization
    for _ in range(max_iter):
        for i in range(n):
            # Generate a new solution via Levy flight
            new_nest = nests[i] + levy_flight(1.5) *
np.random.randn(dim)
            # Apply bounds
            new_nest = np.clip(new_nest, [b[0] for b in
bounds], [b[1] for b in bounds])
            new_fitness = obj_function(new_nest)
            # Update if new solution is better
            if new_fitness < fitness[i]:
                nests[i] = new_nest
```

```
                fitness[i] = new_fitness

        # Abandon some nests and create new ones
        abandon_idx = np.random.rand(n) < pa
        for i in np.where(abandon_idx)[0]:
            nests[i] = np.random.rand(dim) *
(np.array([b[1] for b in bounds]) - np.array([b[0] for b
in bounds])) + np.array([b[0] for b in bounds])
            fitness[i] = obj_function(nests[i])

    # Return the best solution
    best_idx = np.argmin(fitness)
    return nests[best_idx], fitness[best_idx]

# Example usage: Minimize f(x) = x^2
def objective(x):
    return sum(xi**2 for xi in x)

bounds = [(-10, 10), (-10, 10)]  # 2D problem
best_solution, best_fitness = cuckoo_search(objective,
bounds)
print("Best Solution:", best_solution)
print("Best Fitness:", best_fitness)
```

Output:

```
Best Solution: [0.20618515 0.76221114]
Best Fitness: 0.6234781374664989
```