<u>EXP #4 : Cuckoo Search Algorithm</u>

Code:

```
import numpy as np
import math
import random

def levy_flight(d):
   beta = 1.5
   sigma = (np.power(math.gamma(1 + beta), 1/beta) * np.sin(np.pi * beta / 2)) /
np.power(math.gamma((1 + beta) / 2), 1/beta)
   u = np.random.randn(d) * sigma
   v = np.random.randn(d)
   step = u / np.power(np.abs(v), 1/beta)
   return step

def cuckoo_search(fobj, dim, n, pa, itermax):
   # Initialize the population
   X = np.random.rand(n, dim)

   # Iterate until the maximum number of iterations
   for t in range(itermax):
      # Generate new solutions using Lévy flights
      for i in range(n):
         if np.random.rand() < pa:
            X[i] = X[i] + levy_flight(dim)

      # Evaluate the fitness of each solution
      fitness = np.array([fobj(x) for x in X])

      # Rank the solutions
      best_idx = np.argmin(fitness)
      best_sol = X[best_idx]

      # Replace worst solutions with new ones
      worst_idx = np.argmax(fitness)
      X[worst_idx] = best_sol + np.random.rand(dim)

   return best_sol, fitness[best_idx]

# Example usage with a simple objective function
def objective_function(x):
   # Minimize the sphere function
```

```
    return np.sum(x**2)

# Define problem parameters
dim = 10  # Number of dimensions
n = 20   # Population size
pa = 0.25  # Probability of using Lévy flight
itermax = 100  # Maximum number of iterations

# Run Cuckoo Search
best_sol, best_fitness = cuckoo_search(objective_function, dim, n, pa, itermax)

print("Best solution:", best_sol)
print("Best fitness:", best_fitness)
```

Output:

```
Best solution: [-1.41976659  3.92801995  1.40279736  1.4690948  -0.50255297  2.13392306
  2.16805944  2.87217113  1.28548219 -0.23879715]
Best fitness: 41.036682212520766
```