Program 4

Cuckoo Search (CS) Cuckoo Search Algorithms: We need to maximize the total value of selected items without exceeding the knapsack's weight capacity. Using the Cuckoo Search Algorithm, each solution is a binary vector, new solutions are generated via Lévy flights, and the best feasible solution is iteratively improved while abandoning poor solutions with a probability.

Observation:

Cuckoo Search Algorithm

→ Robust Optimization method inspired by the breeding behavior of cuckoo birds.

Step 1: Initialization

n: number of host nests

P(a): Probability of discovering cuckoo's egg

Maxt: Maximum number of iterations to reach optimal solutions

Step 2: Generating Solutions through Levy flight

$$x_i^{t+1} = x_i^t + \alpha \oplus Levy(\lambda)$$

Step 3: Fitness Evaluation

```
If (fitness of CuckooEgg > FitnessOfHostEgg) {
    Replace Host egg with Cuckoo
    t = t+1;
}

if (fitness of CuckooEgg < FitnessOfHostEgg) {
    worst case
    Cuckoo egg killed or thrown away
    new Solution using levy Flight
}
```

Application

Q) Minimize $f(x) = x^2$
global minimum at $x=0$ $f(0) = 0$

Step 1

$x1 = 4$ $f(x1) = 16$
$x2 = -3$ $f(x2) = 9$
$x3 = 6$ $f(x3) = 36$

Best Sol^n
$x2 = -3$, fitness = 9

Step 2

$x_{new} = x_{old} + \alpha \cdot Levy(\lambda)$
$\alpha = 1$
$x_1^{new} = 4 + (-2) = 2$ $f(2) = 4$
$x_2^{new} = -3 + (1) = -2$ $f(-2) = 4$
$x_3^{new} = 6 + (-4) = 2$ $f(2) = 4$

Best sol^n $x = 2$ $f(2) = 4$

Step 3

Evaluate and Select Best

Nest 1 = 2 (fitness 4)
Nest 2 = -2 (fitness 4)
Nest 3 = 2 (fitness 4)

Step 4

$Pa = 0.25$
$x3 = -1$ $f(-1) = 1$
$x = -1$ $f(x) = 1$

Step 5:

$x = 0.5 \rightarrow f(x) = 0.25$

$x = 0 \quad f(x) = 0$

Global minimum found!

Eg

(1) $f(x) = x_1^2 + x_2^2$

Number of nests = 25

discovery rate = 0.25

100 iterations

25 nests in range $[-5, 5]$

Problem: Welded - Beam Design

minimize fabrication cost (material + welding) subject to constraints (stress, buckling, deflection, geometry).

x_1 = h (weld thickness) (in)

x_2 = l (weld length) (inches)

x_3 = t = beam height

x_4 = b = beam width

Minimize $f(x) = 1.10471\, x_1^2 x_2 + 0.04811\, x_3 x_4 (14 + x_2)$

(1) Initialize.

Set parameters n, P_a, T, λ, M.

penalized fitness

$F(x) = f(x) + M \cdot \sum_i \max(0, g_i(x))^2$

record x_{best}.

② loop $t = 1$ to T

Generate new solutions by Levy flights.

③ New candidate: $y = x_i + \alpha \cdot s \odot (x_i - x_{best})$

step scale

elementwise multiply

④ Evaluate candidates

$F(y) < F(x_i)$

select worst $k = \lceil p_\alpha \cdot n \rceil$ nests

⑤ update best.

⑥ ~~Adapti~~ stopping check

stop when $t \geq T$

M.S. 17/10/25

Code:

```
import numpy as np



def f(x):
    return x**2



def levy_flight():
    return np.random.randn()
```

```python
def cuckoo_search(n=3, pa=0.25, iterations=5):
    nests = np.array([4, -3, 6], dtype=float)
    best = nests[np.argmin(f(nests))]

    for t in range(iterations):
        for i in range(n):

            step = levy_flight()
            new = nests[i] + 1 * step


            if f(new) < f(nests[i]):
                nests[i] = new


        for i in range(n):
            if np.random.rand() < pa:
                nests[i] = np.random.uniform(-5, 5)

        best = nests[np.argmin(f(nests))]
        print(f"Iteration {t+1} | Best = {best}, f(x) = {f(best)}")

    return best


best_solution = cuckoo_search()
print("Final Best:", best_solution)
```

Output:

```
Iteration 1 | Best = -3.6431435629308795, f(x) = 13.272495020124703
Iteration 2 | Best = 2.189550165039872, f(x) = 4.794129925226131
Iteration 3 | Best = -0.2714863685552089, f(x) = 0.07370484831129473
Iteration 4 | Best = -0.2714863685552089, f(x) = 0.07370484831129473
Iteration 5 | Best = 0.2459712211369125, f(x) = 0.06050184162758391
Final Best: 0.2459712211369125
```