**LAB 6**

**CUCKOO SEARCH:(USING JOB SCHEDULING)**

import numpy as np

class CuckooSearch:

def \_\_init\_\_(self, jobs, machines, max\_iter=3, population\_size=30, pa=0.25, alpha=0.01, beta=1.5):

self.jobs = jobs

self.machines = machines

self.max\_iter = max\_iter

self.population\_size = population\_size

self.pa = pa

self.alpha = alpha

self.beta = beta

self.num\_jobs = len(jobs)

self.population = np.random.randint(0, machines, (population\_size, self.num\_jobs))

self.fitness = np.array([self.calculate\_makespan(ind) for ind in self.population])

self.best\_solution = self.population[np.argmin(self.fitness)]

self.best\_fitness = np.min(self.fitness)

def calculate\_makespan(self, solution):

machine\_times = [0] \* self.machines

for job, machine in enumerate(solution):

machine\_times[machine] += self.jobs[job]

makespan = max(machine\_times)

return makespan

def levy\_flight(self):

return np.random.normal(0, 1, self.num\_jobs)

def generate\_new\_solution(self, solution):

new\_solution = solution + self.alpha \* self.levy\_flight()

new\_solution = np.clip(new\_solution, 0, self.machines - 1)

new\_solution = np.round(new\_solution).astype(int)

return new\_solution

def run(self):

for iter\_num in range(self.max\_iter):

for i in range(self.population\_size):

new\_solution = self.generate\_new\_solution(self.population[i])

new\_fitness = self.calculate\_makespan(new\_solution)

if new\_fitness < self.fitness[i]:

self.population[i] = new\_solution

self.fitness[i] = new\_fitness

best\_solution\_index = np.argmin(self.fitness)

best\_solution = self.population[best\_solution\_index]

best\_fitness = self.fitness[best\_solution\_index]

if best\_fitness < self.best\_fitness:

self.best\_solution = best\_solution

self.best\_fitness = best\_fitness

for i in range(self.population\_size):

if np.random.rand() < self.pa:

self.population[i] = self.best\_solution

self.fitness[i] = self.best\_fitness

print(f"Iteration {iter\_num + 1}/{self.max\_iter}: Best Makespan = {self.best\_fitness}")

return self.best\_solution, self.best\_fitness

jobs = [10, 20, 30, 40, 50, 60]

machines = 3

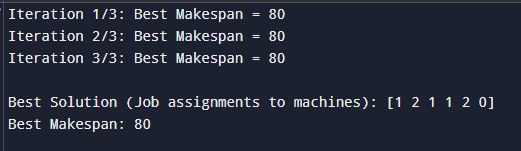
cuckoo = CuckooSearch(jobs, machines, max\_iter=3)

best\_solution, best\_makespan = cuckoo.run()

print("\nBest Solution (Job assignments to machines):", best\_solution)

print("Best Makespan:", best\_makespan)

**OUTPUT:**

****