Name: **Kunal Sinha** Batch: **Swarm R1- G2** Roll no: **2K17/CO/164**

Experiment 5

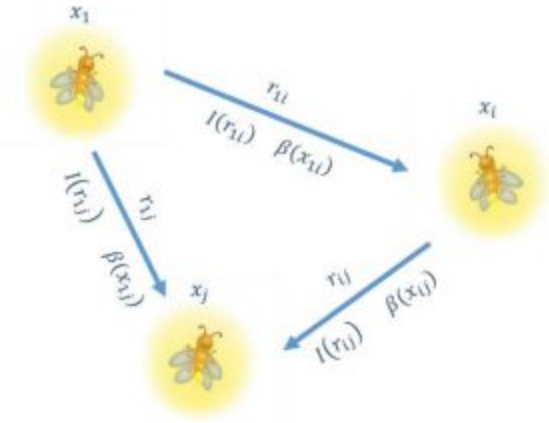
Aim: Write a program to implement Firefly Optimization algorithm.

Theory:

Most species of fireflies are able to glow producing short flashes. It is considered that the main function of flashes is to attract fireflies of the opposite gender and potential prey. Besides, a signal flash can communicate to a predator that a firefly has a bitter taste.

The Firefly optimization Algorithm is based on two important things: the change in light intensity and attractiveness. For simplicity, it is assumed that the attractiveness of a firefly is defined by its brightness which is connected with the objective function.
The algorithm utilizes the following firefly behaviour model:

- All fireflies are able to attract each other independently of their gender;
- A firefly attractiveness for other individuals is proportional to its brightness.
- Less attractive fireflies move in the direction of the most attractive one.
- As the distance between two fireflies increases, the visible brightness of the given firefly for the other decreases.
- If a firefly sees no firefly that is brighter than itself, it moves randomly.



Working of firefly algorithm meta-heuristic function

Algorithm:

1. Objective function f(x), x=(x1, x2, ... , xd)T
2. Initialize a population of fireflies xi(i = 1, 2, ... , n)
3. Define light absorption coefficient gamma
4. WHILE count < Maximum Generations
    a. FOR i = 1 : n (all n fireflies)
        i. FOR j = 1 : i
        ii. Light intensity Ii at xi is determined by f(xi)
        iii. IF Ii > Ij
            1. Move firefly i towards j in all d dimensions
            2. ELSE
            3. Move firefly i randomly
            4. END IF
            5. Attractiveness changes with distance r via exp[-γ r2]
        iv. Determine new solutions and revise light intensity
        v. END FOR j
    b. END FOR i
    c. Rank the fireflies according to light intensity and find the current best
5. END WHILE

Source Code:

firefly.py

```
from math import exp
import numpy as np
from . import intelligence

class fa(intelligence.sw):
    """   Firefly Algorithm """

    def __init__(self, n, function, lb, ub, dimension, iteration, csi=1, psi=1,
                 alpha0=1, alpha1=0.1, norm0=0, norm1=0.1):
        """
        :param n: number of agents
        :param function: test function
        :param lb: lower limits for plot axes
        :param ub: upper limits for plot axes
        :param dimension: space dimension
        :param iteration: number of iterations
        :param csi: mutual attraction
        :param psi: light absorption coefficient of the medium
        :param alpha0: initial value of the free randomization parameter alpha
        :param alpha1: final value of the free randomization parameter alpha
```

```
        :param norm0: first parameter for a normal (Gaussian) distribution
        :param norm1: second parameter for a normal (Gaussian) distribution
        """
        super(fa, self).__init__()
        self.__agents = np.random.uniform(lb, ub, (n, dimension))
        self._points(self.__agents)

        Pbest = self.__agents[np.array([function(x)
                                for x in self.__agents]).argmin()]
        Gbest = Pbest

        for t in range(iteration):
            alpha = alpha1 + (alpha0 - alpha1) * exp(-t)
            for i in range(n):
                fitness = [function(x) for x in self.__agents]
                for j in range(n):
                    if fitness[i] > fitness[j]:
                        self.__move(i, j, t, csi, psi, alpha, dimension,
                                    norm0, norm1)
                    else:
                        self.__agents[i] += np.random.normal(norm0, norm1,
                                                             dimension)

            self.__agents = np.clip(self.__agents, lb, ub)
            self._points(self.__agents)
            Pbest = self.__agents[
                np.array([function(x) for x in self.__agents]).argmin()]
            if function(Pbest) < function(Gbest):
                Gbest = Pbest
        self._set_Gbest(Gbest)

    def __move(self, i, j, t, csi, psi, alpha, dimension, norm0, norm1):
        r = np.linalg.norm(self.__agents[i] - self.__agents[j])
        beta = csi / (1 + psi * r ** 2)

        self.__agents[i] = self.__agents[j] + beta * ( self.__agents[i] - self.__agents[j]) + alpha *
                                exp(-t) * \ np.random.normal(norm0, norm1, dimension)
```

main.py

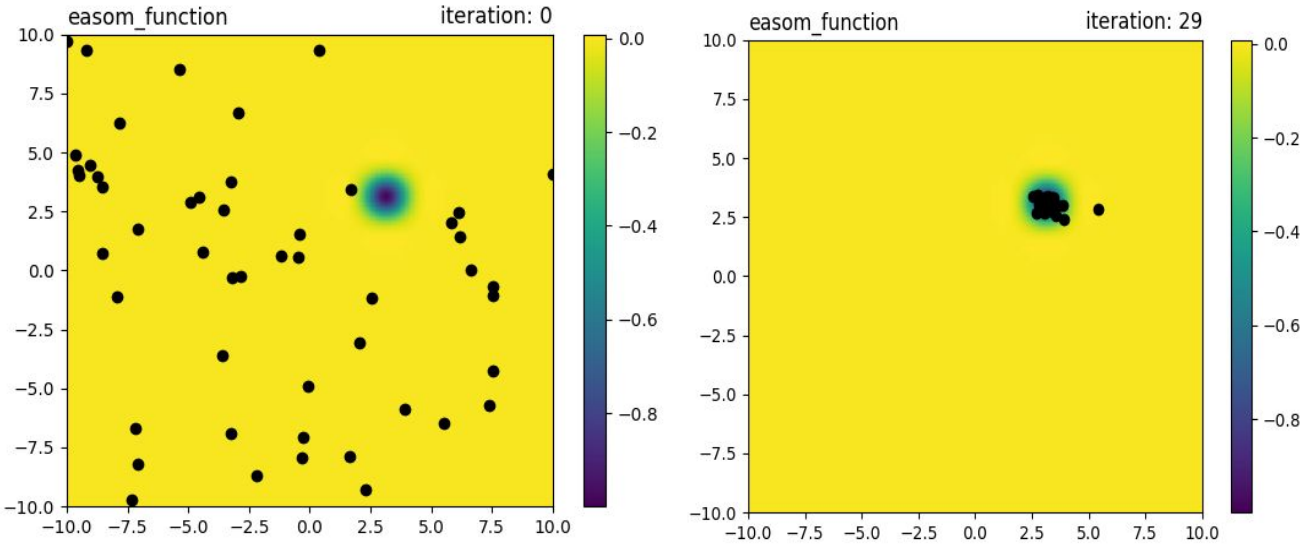
```
import firerfly.py as fa
import matplotlib.pyplot as plt
```

```
def easom_function(x):
    return -cos(x[0])*cos(x[1])*exp(-(x[0] - pi)**2 - (x[1] - pi)**2)

alh = fa(50, easom_function, -10, 10, 2, 30,1,1,1,0.1,0,0.1)
plt(alh.get_agents(),easom_function, -10, 10)
```

Output:

For 30 iterations



Finding and Learnings:

We have successfully implemented the Firefly optimization Algorithm in python. The “firefly algorithm” (FFA) is a modern metaheuristic algorithm, inspired by the behavior of fireflies. This algorithm and its variants have been successfully applied to many continuous optimization problems.