Experiment-6

Implementing the python program for N-Queen Problem using Genetic Algorithm.

Objective:

We have to create a python cod for N-Queen Problem using Genetic Algorithm and then checking the validation of this code by providing different values of N.

N-Queen Problem:

```
import random

import matplotlib.pyplot as plt


n = 8

p = 500


current_generation = []

new_generation = []


def randomGeneration(NumberOfRows,NumberOfQueens):

    generation_list = []

    for i in range(NumberOfRows):

        gene = []

        for j in range(NumberOfQueens):

            gene.append(random.randint(1,n))

        gene.append(0)

        generation_list.append(gene)
```

```
    return generation_list



def fitness(population_list):
    i = 0
    conflict = 0
    while i < len(population_list):
        j = 0
        conflict = 0
        while j < n:
            l = j+1

            while l < n:
                if population_list[i][j] == population_list[i][l]:
                    conflict+=1
                if abs(j-l)==abs(population_list[i][j]-population_list[i][l]):
                    conflict+=1
                l+=1
            j+=1
        population_list[i][len(population_list[j])-1]=conflict
        i+=1

    for i in range(len(population_list)):
        min = i
```

```
        for j in range(i,len(population_list)):
            if population_list[j][n]<population_list[min][n]:
                min = j
        temp =  population_list[i]
        population_list[i] = population_list[min]
        population_list[min] = temp
    return population_list


def cross_over(generation_list):
    for i in range(0,len(generation_list),2):
        z = 0
        new_kid1 = []
        new_kid2 = []
        while z<n:
            if(z<n//2):
                new_kid1.append(generation_list[i][z])
                new_kid2.append(generation_list[i+1][z])
            else:
                new_kid1.append(generation_list[i+1][z])
                new_kid2.append(generation_list[i][z])
            z+=1
        new_kid1.append(0)
        new_kid2.append(0)
        generation_list.append(new_kid1)
```

```
        generation_list.append(new_kid2)
    return generation_list



def mutation(generation_list):
    muted_list=[]
    i = 0
    while i<p//2:
        new_rand = random.randint(p//2,p-1)
        if new_rand not in muted_list:
            muted_list.append(new_rand)
            generation_list[new_rand][random.randint(0,n-
1)]=random.randint(1,n-1)

            i+=1
    return generation_list


def showRes(res):
    l = len(res)
    plt.figure(figsize=(6, 6))
    plt.scatter([x+1 for x in range(l - 1)], res[:l - 1])
    for i in range(l):
        plt.plot([0.5, l - 0.5], [i + 0.5, i + 0.5], color = "k")
        plt.plot([i + 0.5, i + 0.5], [0.5, l - 0.5], color = "k")
```

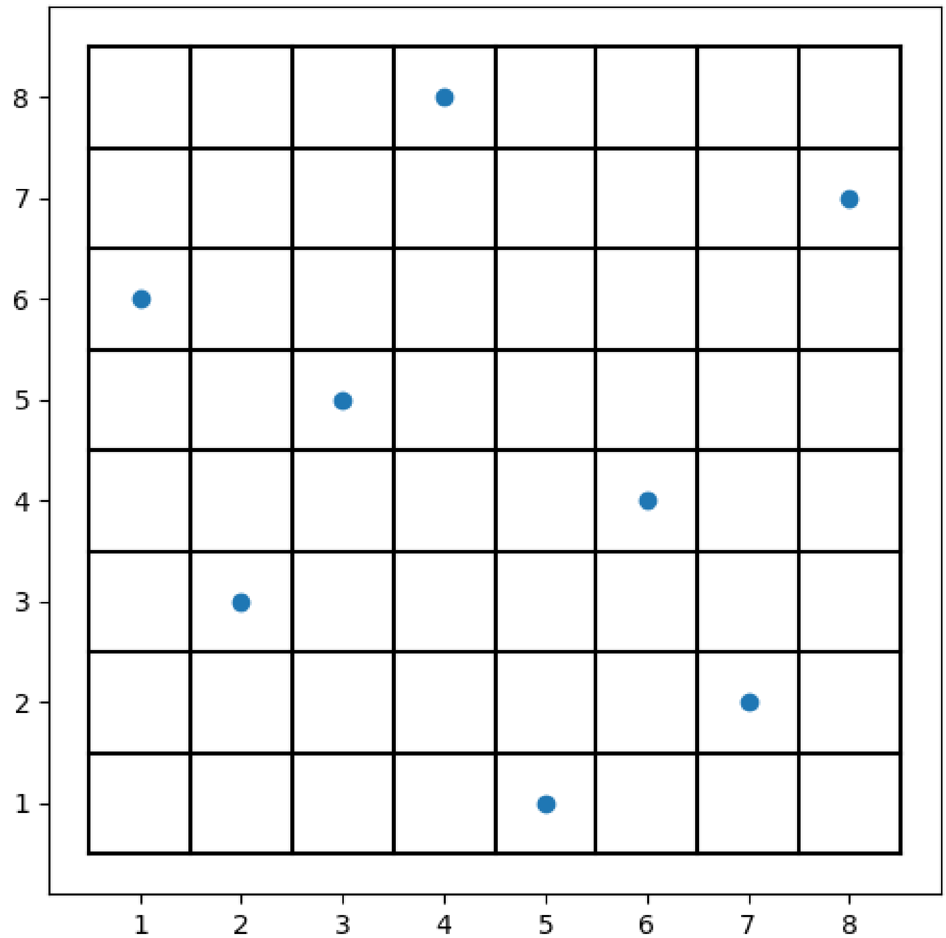
```
    plt.show()



current_generation = randomGeneration(p,n)
current_generation = fitness(current_generation)
epoch = 1
while True:
    print("------------------------------------------------------")
    print("Step ",epoch)
    current_generation = current_generation[0:p//2]
    new_generation = cross_over(current_generation)
    new_generation = mutation(new_generation)
    current_generation = new_generation
    current_generation = fitness(current_generation)
    if current_generation[0][n] == 0:
        print("Solution Found: ", current_generation[0])
        showRes(current_generation[0])
        break
    else:
        print("Best Solution: ", current_generation[0])
    epoch+=1
```

Output:

```
======= RESTART: U:\4-1\Artificial Intelligence LAB\Lab6
------------------------------------------------------------
Step  1
Best Solution:  [8, 5, 7, 1, 6, 2, 5, 3, 2]
------------------------------------------------------------
Step  2
Best Solution:  [8, 5, 7, 1, 6, 2, 5, 3, 2]
------------------------------------------------------------
Step  3
Best Solution:  [8, 5, 7, 1, 6, 2, 5, 3, 2]
------------------------------------------------------------
Step  4
Best Solution:  [8, 5, 7, 1, 6, 2, 5, 3, 2]
------------------------------------------------------------
Step  5
Best Solution:  [8, 5, 7, 1, 6, 2, 5, 3, 2]
------------------------------------------------------------
Step  6
Best Solution:  [8, 5, 7, 1, 6, 2, 5, 3, 2]
------------------------------------------------------------
Step  7
Best Solution:  [8, 5, 7, 1, 6, 2, 5, 3, 2]
------------------------------------------------------------
Step  8
Best Solution:  [3, 1, 6, 2, 5, 1, 8, 4, 1]
------------------------------------------------------------
Step  9
Best Solution:  [3, 1, 6, 2, 5, 1, 8, 4, 1]
------------------------------------------------------------
Step  10
Best Solution:  [3, 1, 6, 2, 5, 1, 8, 4, 1]
------------------------------------------------------------
Step  11
Best Solution:  [3, 1, 6, 2, 5, 1, 8, 4, 1]
------------------------------------------------------------
Step  12
Solution Found:  [6, 3, 5, 8, 1, 4, 2, 7, 0]
```


8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8