

Genetic Algorithm & Fuzzy Logic

Semester-5

Practical - 6

Name: Shivam Tawari

Roll no: A-58

Aim: Implement Traveling Salesman Problem (TSP) solving using Genetic Algorithm.

Theory:

Approach:

Gene: a city (represented as (x, y) coordinates)

Individual: a single route satisfying the conditions above

Population: a collection of possible routes

Parents: two routes that are combined to create a new route

Mating pool: a collection of parents that are used to create our next population

Fitness: a function that tells us how good each route is

Mutation: a way to introduce variation in our population by randomly swapping two cities in a route

City:

We first create a City class that will allow us to create and handle our cities. These are simply our (x, y) coordinates.

Fitness:

We'll also create a Fitness class. In our case, we'll treat the fitness as the inverse of the route distance. We want to minimize route distance, so a larger fitness score is better.

Population:

We now can make our initial population. To do so, we need a way to create a function that produces routes that satisfy our conditions. To create an individual, we randomly select the order in which we visit each city.

Fitness:

The fitness function simply defined is a function which takes a candidate solution to the problem as input and produces as output how “fit” our how “good” the solution is with respect to the problem in consideration.

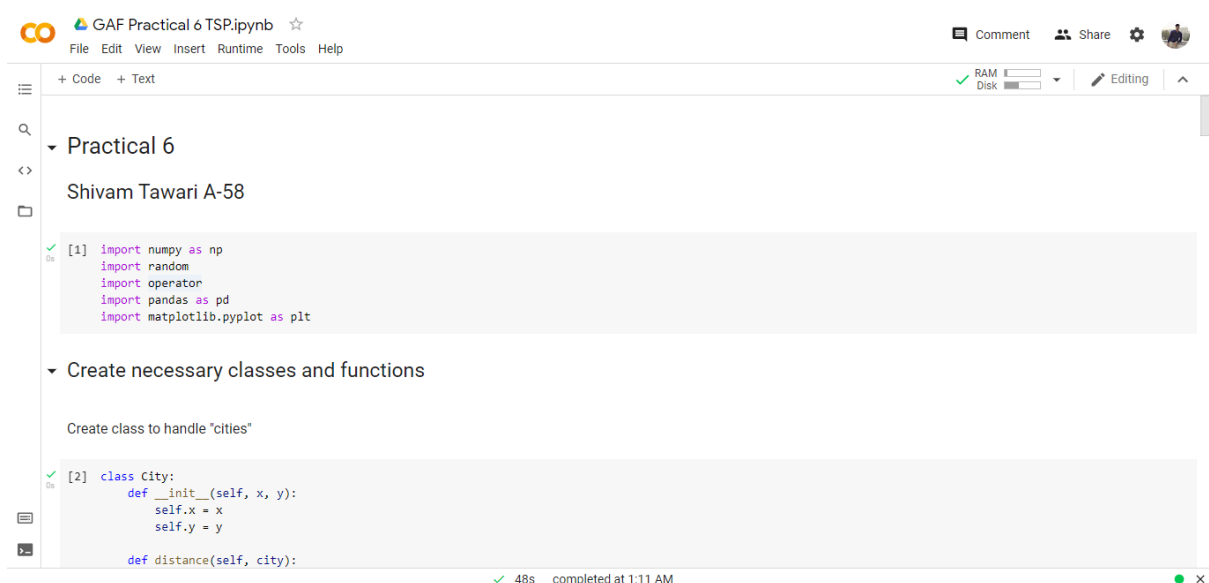
Mating Pool:

There are a few options for how to select the parents that will be used to create the next generation. The most common approaches are either fitness proportionate selection or tournament selection:

Fitness proportionate selection (the version implemented below): The fitness of each individual relative to the population is used to assign a probability of selection. Think of this as the fitness-weighted probability of being selected.

Tournament selection: A set number of individuals are randomly selected from the population and the one with the highest fitness in the group is chosen as the first parent. This is repeated to choose the second parent.

Code:



```
[1] import numpy as np
import random
import operator
import pandas as pd
import matplotlib.pyplot as plt

[2] class City:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self, city):
```


CO GAF Practical 6 TSP.ipynb ☆
File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

RAM  Disk  Editing

Create list of cities

```
[15] cityList = []  
  
for i in range(0,25):  
    cityList.append(City(x=int(random.random() * 200), y=int(random.random() * 200)))
```

Run the genetic algorithm

```
geneticAlgorithm(population=cityList, popSize=100, eliteSize=20, mutationRate=0.01, generations=500)
```

Initial distance: 2159.9824004068
Final distance: 833.8244491105012
[(156,59),
 (182,66),
 (176,73),
 (135,81),
 (112,117),
 (109,122),
 (111,142),
 (118,161),
 (138,164),
 (140,149),
 (198,146),
 (172,192),
 (132,104)]

48s completed at 1:11 AM

CO GAF Practical 6 TSP.ipynb ☆
File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

RAM  Disk  Editing

```
[16] geneticAlgorithm(population=cityList, popSize=100, eliteSize=20, mutationRate=0.01, generations=500)
```

Initial distance: 2159.9824004068
Final distance: 833.8244491105012
[(156,59),
 (182,66),
 (176,73),
 (135,81),
 (112,117),
 (109,122),
 (111,142),
 (118,161),
 (138,164),
 (140,149),
 (198,146),
 (172,192),
 (133,181),
 (136,192),
 (87,182),
 (76,179),
 (0,171),
 (13,157),
 (27,153),
 (40,45),
 (18,25),
 (40,16),
 (41,12),
 (114,39),
 (139,64)]

48s completed at 1:11 AM

Conclusion: Hence, Travelling Sales Problem (TSP) using GA has been implemented successfully.