Experiment No: 10

Objective: Solve the Traveling Salesman Problem with Genetic Algorithm.

Theory: The Traveling Salesman Problem (TSP) is a well-known optimization problem that aims to find the shortest possible route that visits all cities exactly once and returns to the starting city. Since it is an NP-hard problem, finding the exact solution becomes impractical for large numbers of cities.

To overcome this, the Genetic Algorithm (GA) is used as an efficient heuristic approach. GA is inspired by the process of natural selection, where a population of possible solutions evolves toward better ones over generations. It uses key operations such as selection, crossover, and mutation to explore and exploit the solution space. In TSP, each route is represented as a chromosome, and the fitness function evaluates the total distance of the tour. Through repeated evolution, GA converges to an optimal or near-optimal route efficiently.

Source Code:

Traveling_Salesman.py U ×

Traveling_Salesman.py > ...

```python
def calculate_result(Population):
            column = individual[i+1]
            distance += Distance_Matrix[row][column]

        if distance < minimum:
            minimum = distance
            shortest_path = individual

    return shortest_path, minimum

Population = initial_population()

for gen in range(Generations):
    print(f'\nGeneration: {gen+1}\n')

    Fitness = fitness_calculation(Population)

    for i in range(len(Population)):
        print(f'Population {i+1}: {Population[i]} --> Fitness: {Fitness[i]}')

    Ancestor_1_Index, Ancestor_2_Index = selection_roulette_wheel(Fitness)

    print(f'\nAncestor {Ancestor_1_Index+1}:\t\t {Population[Ancestor_1_Index]}')
    print(f'Ancestor {Ancestor_2_Index+1}:\t\t {Population[Ancestor_2_Index]}')

    Descendant = crossover(Population[Ancestor_1_Index], Population[Ancestor_2_Index])
    Mutated_Descendant = mutation(Descendant)

    print(f'Descendant:\t\t {Descendant}')

    New_Generation = new_generation(Population, Fitness, Mutated_Descendant)
    Population = list(New_Generation)
    Current_Best_Result = calculate_result(Population)

    print(f'\nCurrent Generation Result\nShortest Circular Tour: {Current_Best_Result[0]}')
    print(f'Total Travel Cost: {Current_Best_Result[1]}\n')
```

Source Code On GitHub:
https://github.com/MdImranKhanSiam/Problem_Solving/blob/main/Practice/All_Program/Python/Traveling_Salesman.py

Algorithm:

1. Start
2. Initialize parameters:
    - Number of cities
    - Population size
    - Number of generations
    - Distance matrix between cities
3. Generate the initial population:
    - Randomly create several tours (chromosomes) representing possible city orders.
    - Each tour starts and ends at the same city.
4. Calculate fitness for each chromosome:
    - Compute the total travel distance for each tour.
    - Fitness = 1 / (total distance).
5. Select parents (ancestors):
    - Apply roulette wheel selection based on fitness values to choose two parents for crossover.
6. Apply crossover:
    - Select a random segment from the first parent.
    - Fill remaining cities from the second parent while maintaining city order.
    - Form a new descendant (offspring).
7. Apply mutation:
    - Randomly swap two cities in the descendant with a small probability.
8. Form new generation:
    - Insert the mutated descendant into the population.
    - Remove the least fit individual (natural selection).
9. Repeat steps 4–8 for the specified number of generations.
10. Evaluate the final population:
    - Identify the chromosome with the shortest total travel distance.
11. Display results:
    - Show the best tour and its total travel cost.
12. End

Sample Input/Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\IMRAN\UGV\University Of Global Village\7th Semester\Courses\CSE 406 - Artifi
alesman.py

Generation: 1

Population 1: [0, 4, 1, 3, 2, 0] --> Fitness: 0.0030864197530864196
Population 2: [4, 3, 2, 0, 1, 4] --> Fitness: 0.0028169014084507044
Population 3: [3, 4, 0, 1, 2, 3] --> Fitness: 0.004524886877828055
Population 4: [3, 2, 0, 4, 1, 3] --> Fitness: 0.0030864197530864196
Population 5: [4, 3, 2, 0, 1, 4] --> Fitness: 0.0028169014084507044
Population 6: [1, 0, 2, 3, 4, 1] --> Fitness: 0.006289308176100629

Ancestor 3:              [3, 4, 0, 1, 2, 3]
Ancestor 1:              [0, 4, 1, 3, 2, 0]
Ancestor Segment 3-5:    [0, 1, 2]
Descendant:              [4, 3, 0, 1, 2, 4]
Mutated Descendant:      [4, 3, 1, 0, 2, 4] --> Fitness: [0.005952380952380952]

Natural Selection: Population 2 --> [4, 3, 2, 0, 1, 4]

Current Generation Result
Shortest Circular Tour: [1, 0, 2, 3, 4, 1]
Total Travel Cost: 159


Generation: 2

Population 1: [0, 4, 1, 3, 2, 0] --> Fitness: 0.0030864197530864196
Population 2: [3, 4, 0, 1, 2, 3] --> Fitness: 0.004524886877828055
Population 3: [3, 2, 0, 4, 1, 3] --> Fitness: 0.0030864197530864196
Population 4: [4, 3, 2, 0, 1, 4] --> Fitness: 0.0028169014084507044
Population 5: [1, 0, 2, 3, 4, 1] --> Fitness: 0.006289308176100629
Population 6: [4, 3, 1, 0, 2, 4] --> Fitness: 0.005952380952380952

Ancestor 6:              [4, 3, 1, 0, 2, 4]
Ancestor 2:              [3, 4, 0, 1, 2, 3]
Ancestor Segment 3-5:    [1, 0, 2]
Descendant:              [3, 4, 1, 0, 2, 3]
Mutated Descendant:      [3, 4, 1, 0, 2, 3] --> Fitness: [0.006289308176100629]

Natural Selection: Population 4 --> [4, 3, 2, 0, 1, 4]
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\IMRAN\UGV\University Of Global Village\7th Semester\Courses\CSE 406 - Artifi
alesman.py
Population 4: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629
Population 5: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629
Population 6: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629

Ancestor 6:              [3, 4, 1, 0, 2, 3]
Ancestor 2:              [3, 4, 1, 0, 2, 3]
Ancestor Segment 1-4:    [3, 4, 1, 0]
Descendant:              [3, 4, 1, 0, 2, 3]
Mutated Descendant:      [3, 0, 1, 4, 2, 3] --> Fitness: [0.005050505050505051]

Natural Selection: Descendant --> [3, 4, 1, 0, 2, 3]

Current Generation Result
Shortest Circular Tour: [3, 4, 1, 0, 2, 3]
Total Travel Cost: 159


Generation: 46

Population 1: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629
Population 2: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629
Population 3: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629
Population 4: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629
Population 5: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629
Population 6: [3, 4, 1, 0, 2, 3] --> Fitness: 0.006289308176100629

Ancestor 2:              [3, 4, 1, 0, 2, 3]
Ancestor 6:              [3, 4, 1, 0, 2, 3]
Ancestor Segment 1-3:    [3, 4, 1]
Descendant:              [3, 4, 1, 0, 2, 3]
Mutated Descendant:      [3, 4, 1, 0, 2, 3] --> Fitness: [0.006289308176100629]

Natural Selection: Population 1 --> [3, 4, 1, 0, 2, 3]

Current Generation Result
Shortest Circular Tour: [3, 4, 1, 0, 2, 3]
Total Travel Cost: 159

PS C:\IMRAN\UGV\University Of Global Village\7th Semester\Courses\CSE 406 - Artifi
```