USN: 1BM22CS259

LAB-6: Simulated Annealing Algorithm for:

1) 8 queens problem

CODE:

```
import mlrose_hiive as mlrose

import numpy as np


def queens_max(position):

    no_attack_on_j = 0

    queen_not_attacking = 0

    for i in range(len(position) - 1):

        no_attack_on_j = 0

        for j in range(i + 1, len(position)):

            if (position[j] != position[i]) and (position[j] != position[i] + (j - i)) and (position[j] != position[i] -
(j - i)):

                no_attack_on_j += 1

        if (no_attack_on_j == len(position) - 1 - i):

            queen_not_attacking += 1

    if (queen_not_attacking == 7):

        queen_not_attacking += 1

    return queen_not_attacking


def print_board(position):

    size = len(position)

    board = np.full((size, size), '.')

    for row, col in enumerate(position):

        board[row, col] = 'Q'

    print('\n'.join([' '.join(row) for row in board]))


objective = mlrose.CustomFitness(queens_max)
```

```
problem = mlrose.DiscreteOpt(length=8, fitness_fn=objective, maximize=True, max_val=8)

T = mlrose.ExpDecay()


initial_position = np.array([4, 6, 1, 5, 2, 0, 3, 7])


best_position, best_objective, fitness_curve = mlrose.simulated_annealing(problem=problem,
schedule=T, max_attempts=500, init_state=initial_position)


print('The best position found is:', best_position)

print('The number of queens that are not attacking each other is:', best_objective)


print("Board representation:")

print_board(best_position)
```

OUTPUT:

```
The best position found is: [2 5 7 0 3 6 4 1]
The number of queens that are not attacking each other is: 8.0
Board representation:
. . Q . . . . .
. . . . . Q . .
. . . . . . . Q
Q . . . . . . .
. . . Q . . . .
. . . . . . Q .
. . . . Q . . .
. Q . . . . . .
```

2) Travelling Salesman Problem

Code:

```
import mlrose_hiive as mlrose

import numpy as np

from scipy.spatial.distance import euclidean


# Define the coordinates of the cities

coords = [(0, 0), (1, 5), (2, 3), (5, 1), (6, 4), (7, 2)]
```

```
# Calculate the distances between each pair of cities
distances = []
for i in range(len(coords)):
    for j in range(i + 1, len(coords)):
        dist = euclidean(coords[i], coords[j])
        distances.append((i, j, dist))
fitness_dists = mlrose.TravellingSales(distances=distances)
problem = mlrose.TSPOpt(length=len(coords), fitness_fn=fitness_dists, maximize=False)
schedule = mlrose.ExpDecay(init_temp=10, exp_const=0.005, min_temp=1)
result = mlrose.simulated_annealing(problem, schedule=schedule, max_attempts=100, max_iters=1000, random_state=2)
print("Result structure:", result)
if isinstance(result, tuple) and len(result) == 2:
    best_state, best_fitness = result
else:
    best_state, best_fitness = result[0], result[1]
print("Best route found:", best_state)
print("Total distance of best route:", best_fitness)
```

OUTPUT:

```
Result structure: (array([1, 0, 3, 5, 4, 2]), 21.0293485853026, None)
Best route found: [1 0 3 5 4 2]
Total distance of best route: 21.0293485853026
```