WriteUp

Assignment 7

Travelling Salesman Problem

Overview:

The **Travelling Salesman Problem (TSP)** involves finding the shortest route that visits every city once and returns to the start. Since this problem is complex, we used **Simulated Annealing (SA)**, a method that helps find a good solution by exploring different routes and slowly improving them over time.

Approach:

1. **Initial Tour**: The process starts with a simple route. We use the **nearest neighbor** approach, where the next city chosen is the closest one to the current city.

2. **Neighboring Solutions**: We create a new solution by swapping two cities in the current route. This is done using the function `get_neighbor_tour()`.

3. **Acceptance of New Solutions**: We compare the cost of the new solution with the current one. If the new solution is better, or if it's accepted randomly, we update the current solution. The chance of accepting a worse solution is controlled by a temperature that gradually decreases over time, making the algorithm focus on better solutions as it progresses. This is handled by the `acceptance_probability()` function.

4. **Cooling Schedule**: At the beginning, the temperature is high, which allows for more exploration of different routes. As the algorithm continues, the temperature lowers, making the system settle into the best possible solution.

5. **Output**: The program saves intermediate results, such as the current route, its cost, and the temperature at each step, into a file. The final best route and its cost are also saved.

Variables and Functions:

- **Variables**:
    - *`dist_matrix`*: A table of distances between the cities.
    - *`start_city`:* The city where the journey starts.
    - *`init_temp`, `cooling_rate`, `iterations`:* Settings for the simulated annealing process.
    - *`curr_tour`, `best_tour`*: The current route and the best route found so far.
    - *`curr_cost`, `best_cost`:* The cost (total distance) of the current and best routes.
- **Functions**:
    - *`calculate_tour_cost()`:* Calculates the total distance of a given route.
    - *`get_neighbor_tour()`:* Creates a neighboring route by swapping two cities.
    - *`acceptance_probability()`:* Decides if a new route should be accepted based on its cost and the current temperature.
    - *`solve_tsp_simulated_annealing()`*: Runs the simulated annealing algorithm, improving the route over many iterations.

Process:

1. **Input**: The program first loads data from an input file, including the distance matrix, starting city, and settings like the initial temperature, cooling rate, and the number of iterations.

2. **Initialization**: We use the nearest neighbor approach to create the first route and calculate its total distance.

3. **Simulated Annealing Loop**: In each iteration:

    - A new route is created by swapping two cities.

    - The cost of the new route is calculated.

    - The new route is accepted based on the cost and a probability related to the temperature.

    - The temperature decreases after each iteration.

4. **Output**: The program writes intermediate results (current route, cost, and temperature) into a file, and at the end, it saves the best route found.

Conclusion:

Simulated Annealing is a powerful method for solving the TSP by balancing exploration and refinement of solutions. While the solution may not always be perfect, it offers a very good approximation, especially for large numbers of cities. The gradual cooling process ensures that the solution becomes better over time, avoiding getting stuck in suboptimal solutions.