Travelling Salesman Problem - Assignment 6 Document

EE22B175

Traveling Salesman Problem (TSP)

The traveling salesman problem (TSP) is a combinatorial optimization problem in which a salesman must visit a set of cities and return to the starting city, minimizing the total distance travelled.

Approach

The approach used to solve the TSP in the code above is a combination of nearest neighbour search and simulated annealing.

Nearest Neighbour Search

Nearest neighbour search is a simple and efficient algorithm for finding a tour of a set of cities. It works by starting at a random city and then repeatedly adding the nearest unvisited city to the tour until all cities have been visited.

Simulated Annealing

Simulated annealing is a metaheuristic algorithm for finding global optima in complex search spaces. It works by iteratively exploring the search space and accepting new solutions with a probability that depends on a temperature parameter. The temperature parameter is gradually decreased over time, which allows the algorithm to escape from local optima and find better solutions.

Usage

To use this script, you would run it from the command line with the filename of the input data as an argument, as shown below

```
python Assignment6.py <filename>
```

The script will read the city coordinates, perform the Nearest Neighbour search, optimize the tour using Simulated Annealing, and display the results, including the percentage improvement.

Implementation

The Python code above implements the following steps to solve the TSP:

1. **Read the city coordinates from a file.**
2. **Find the nearest neighbour search of the cities.**
3. **Use the nearest neighbour search order as the starting point for simulated annealing.**
4. **Iteratively explore the search space by swapping two random cities in the order.**
5. **Accept new distances if they are better than the current distance.**
6. **Accept new solutions with a worse total distance with a probability that depends on the temperature parameter and the difference in total distances between the current and the new order.**
7. **Gradually decrease the temperature parameter over time.**
8. **Return the best order found after the same order is encountered 2000 times in a row.**

Output

The script will output the following information:

- The final order for the salesman to travel.
- The total distance to be travelled along that path.
- The percentage improvement in the path compared to a random initial order.

Additionally, it will display a 2D plot of the cities and the optimized tour.

Improvement Metric

The percentage improvement in the path that is seen starting from a random initial point to the best possible solution found is calculated as follows:

$$improvement = \frac{RandomDistance - BestDistance}{(RandomDistance)} \times 100\%$$

The improvement percentage observed after applying the Nearest Neighbour Search and then Simulated Annealing is between 69% and 72%.

Limitations

- It can be slow to converge to the global optimum.
- It is sensitive to the choice of cooling schedule and neighbourhood search method.
- It may not be effective for finding solutions to large and complex problems.

To address the limitations of the simulated annealing algorithm, the following can be done:

- Use a hybrid algorithm that combines simulated annealing with other algorithms.
- Use a more sophisticated cooling schedule and neighbourhood search method.
- Use a parallel computing framework to speed up the algorithm's convergence.
- The random swap neighbourhood search method used in the code above is simple and efficient, but it may not be the most effective way to explore the search space.
- More efficient neighbourhood search methods, such as the 2-opt or 3-opt methods, can be used to improve the algorithm's performance.
- A logarithmic instead of linear cooling rate (decay in Temperature) could be used to better escape local minima.