MARMARA UNIVERSITY

CSE2246 – HOMEWORK 2

PROJECT REPORT

Sinan Tan – 150122823

Yavuz Selim Okumuş – 150119705

Melih Sinan Çubukçuoğlu – 150122828

Division of Labor

All the group members were active in the project. We have talked with each other in voice chat and did it together.

Introduction

The Two Traveling Salesmen Problem (2 TSP) is a version of the classic Traveling Salesman Problem (TSP). In this variation the goal is to plan two routes, for two salesmen with each route starting and ending in cities. The objective is to ensure that every city is visited once by one of the salesmen while minimizing the total distance traveled. Due to the nature of this problem heuristic methods are essential for finding optimal solutions within a reasonable timeframe.

Our strategy for tackling the 2 TSP consists of stages;

1. **Parsing Input:** Extracting and organizing city coordinates from an input file.
2. **City Assignment:** Randomly dividing cities between the two salesmen.
3. **Tour Creation:** Establishing tours using a neighbor approach.
4. **Tour Enhancement:** Refining the tours using a 2 opt optimization method.
5. **Result Presentation:** Saving the solution in an output file.

In terms of algorithm specifics and data structures;

> The input file contains city information, with IDs and coordinates formatted as "ID x coordinate y coordinate". The information is saved in a collection of City objects each, with the citys identifier and coordinates (x, y).

Dividing Cities (Splitting)

To ensure that each city is visited by one salesperson the cities are divided randomly into two groups. This is accomplished by mixing up the list of cities and assigning them alternately to each group.

Building Tours

Initial tours are created using the neighbor method. Starting from a city chosen at random the algorithm adds the city to the tour until all cities, in the group are included. This provides a good starting point for further optimization.

Optimization for tour

To better the original tours that had been made, we adopt the 2-opt optimization method. This algorithm goes through the tour in a way that it examines pairs of edges and then swaps them provided by swapping would make total distance smaller. Moreover, this local search technique eliminates unnecessary detours, thus reducing the overall travel distance.

Generation of Output

Finally, these last tours are formatted and written to an output file with their distances. The output consists of total distance travelled by each salesman, individual tours' distances and sequence of cities visited in each tour.

Data Structures Used

- City Class: This class represents the features of a city inclusive of its Id and coordinates (x, y) and has a method that can be used to compute a Euclidean distance to another city. The equation for finding the distance is as follows:

$$d(c_1, c_2) = round\left(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\right)$$

  This class is very important since it helps in organizing city data and enabling the measurement of distances between cities.

- List<City>: This data structure is utilized for keeping lists of cities and their corresponding tours conducted by each salesperson. It can adjust size dynamically, making it easier to work with lists of cities and traverse different tours while creating or refining them.

- Set<City>: A Set is employed for tracking visited cities during tour construction phase. One significant advantage of using a Set is that it allows one to quickly check if any element exists thus avoiding repetition of any visited city. This ensures that there are no duplicates in subsequent tours made when constructing the tour's plan.

Conclusion

Our algorithm tackles 2-TSP effectively using heuristics. Our solution works well because we divide the cities into groups for which initial tours are constructed using nearest neighbor heuristic and further optimized with the 2-opt method making it both pragmatic and effective. While not always producing optimal solutions, this computer-intensive system gives nearly optimal results in a reasonable amount of time thereby making it fit large-scale instances of 2