CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean
Webpage: `http://www.cs.clemson.edu/~bcdean/`
Handout 13: Lab #8

Fall 2019
MW 2:30-3:45
Daniel 313

1 Iterative Refinement and the Traveling Salesman Problem (TSP)

In this lab exercise, we will use iterative refinement to build a reasonable good solution for the traveling salesman problem (recall that this is an NP-hard problem, so finding an guaranteed-optimal solution is believed to be computationally intractable). In the file

`/group/course/cpsc212/f19/lab08/tsp_points.txt`

you will find 50 points in the 2D plane on which you should build your TSP tour. Each point is specified by its x and y coordinates, with one point per line. Here is an example of a reasonably good TSP tour on these points:

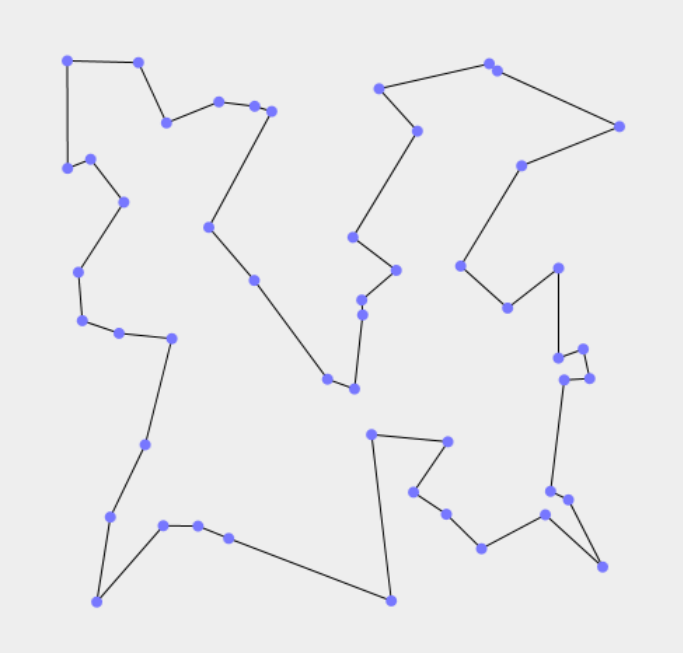

Figure 1: Example TSP solution on 50 cities in the 2D plane.

You will construct your tour using iterative refinement, starting with a random tour that you get by taking a random permutation of an array containing $0 \ldots 49$. There are several ways to randomly

permute the contents of an array $A[0 \ldots N-1]$; the easiest is probably to loop through the array and swap each element with randomly-chosen preceding element.

To refine your tour, you will look at all tours in a local neighborhood obtained by picking two edges in the tour and replacing them with their corresponding "diagonals", as shown in below:

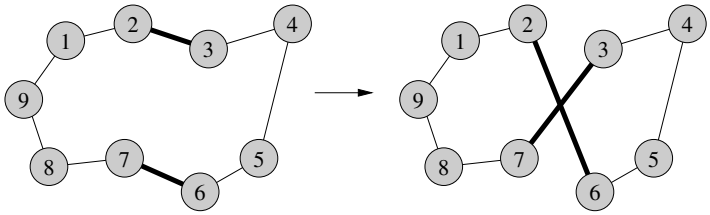

Figure 2: Generating a neighboring solution.

Observe that, as explained in lecture, this operation of swapping two edge for their diagonals corresponds to the reversal of a contiguous subarray in our array of cities. In the example above, the part of the tour that visited cities 3, 4, 5, 6 is now reversed, visiting the same cities in the order 6, 5, 4, 3. In terms of the entire array of all cities, we have changed 1, 2, 3, 4, 5, 6, 7, 8, 9 into 1, 2, 6, 5, 4, 3, 7, 8, 9, having reversed the appropriate subarray.

With 50 cities (and hence 50 edges in the tour), you will have $\binom{50}{2} = 50(49)/2$ neighboring solutions to generate and check. When you find a better neighboring solution, you can either move to this solution immediately, or you can continue searching the entire neighborhood, moving afterward to the best neighboring solution you have found. You should continue refining the solution until it becomes locally optimal. Repeat this entire process from multiple random initial tours to alleviate the problem of getting stuck at poor locally-optimal solutions.

Your code should read its input from `tsp_points.txt` and print as output the length of the best tour it finds, along with the cities in this tour as a sequence of space-separated integers (i.e., a permutation of $0 \ldots 49$).

The website `http://www.cs.clemson.edu/~bcdean/tsp.html` allows you to visualize the tour you produce as output, to help with debugging.

2 Grading

For this lab, you will receive 8 points for correctness, and 2 points for well-written code. Zero points will be awarded for code that does not compile, so make sure your code compiles on the lab machines before submitting!

Final submissions are due by 11:59pm on the evening of Friday, November 1. No late submissions will be accepted.