Document for week 6

Naho Yokoyama

Overview

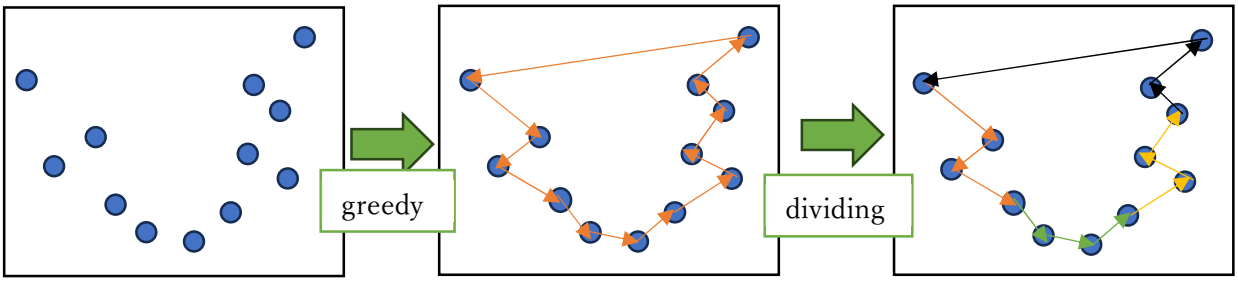
This is a program that solves the TSP. One big difference from the week 5's program is that since I have to deal with inputs with large size, I modified the program so that it divides the list of cities into four sections to do methods like 2-opt. Moreover, I wrote week5's program in python but this time, I rewrite this all in c++ to improve execution time.

Why I rewrite all of my programs in c++

The reason why I rewrite all of my programs in c++ is that by looking at the execution time for python program and c++ program, even if the program itself doing the same thing, c++ program can execute faster. Since I am managing the execution time of functions (simulated annealing), it would be beneficial if I can reduce the execution time of one trial of function because then I can repeat more trials.

How I divide cities into sections

Firstly, I cannot just separate the list of cities because I want to divide so that points in each sector are closed together. Therefore, I first tried greedy methods to come up with the path connecting closest points and divide the sections according to the order of the path from the greedy method. The reason why I chose to divide this into four is that the length of the largest input was 8192 while the second largest was 2048, which is four times difference. Since the execution time of week5's program for the second largest input was reasonable, I thought if I can divide the largest input data to four, the size will be roughly the same as the second largest input and hence will have reasonable execution time when I applied "removeCross" function after that. Also, just in case, I modified the cluster size into 8 and 16, but the score did not improved, maybe because if I increase the cluster size we have less pairs of points for "removeCross" function.



Result

Before I implemented dividing, the length of the execution for the input_7.csv file was very long so that I cannot wait enough for it to finish processing. However, after the implementation, I was able to reduce the execution time for about several minutes (the final length depends on what value did I set for the time limit for simulated annealing)

Things I also considered

Although it did not work, I tried other methods, looking at other students' task. Firstly, I modified my simulated annealing part (especially for the function for finding probability and temperature management) because by discussing with a mentor, this part may be the reason why my simulated annealing part did not work. Therefore, I look at other students (Noda-san, Yamamoto-san) and see what they are doing for annealing. Also, I tried modification of different parameters because a small change in one parameter (such as initial temperature) may change the result of the simulated annealing rapidly. However, there were no significant improvements; the score get worse, or, increased but within a range of random errors.