CMPE160 PROJECT3

SOLVING TSP

SPECIFICATION:

In this project you are going to implement five different solution strategies for the well known travelling salesman problem (TSP). TSP is the problem of finding the shortest tour that covers all given cities such that the tour starts and ends at the same city. The input file for the coordinates of the cities will be given at course website. At each line of the file, the first entry is the ID of the city, the second one is the x coordinate, and the third one is the y coordinate. The tour will start at city 1, and end at the same city. The solution strategies that you are going to implement are as follows:

1. Exact Enumeration: In this strategy, you will try all possible tour combinations and find the shortest one. Let n be a given parameter that denotes the number of cities. For instance, if n=20 you will only consider the first 20 cities in the solution, even though there are 101 cities in the file. Since this takes too much time you will do this for small n. Start with n=6 and increase n by 2 each time (n=8, n=10, …) until it takes too much time to solve the problem (a day or two). Report the results and time passed executing the code (in seconds). Furthermore, write the best result to a file named opt_tsp_n.txt where n is the value of n for that case. The content of the file should be as follows (example for n=8):
   1453 //length of the tour
   1-6-7-8 -2-5-4-3-1 //tour
2. Greedy Method: In this case, the cities will be divided to two sets: Connected and NotConnected. At each step, you will add a city from the set NotConnected that is closest to the city that is most recently added to the set Connected. At first, Connected will only include the first city. The algorithm will run until there are no cities in NotConnected. You will run this method for n values you tried at the first method and also for n=40, n=60, n=80, n=101. Report the results and time passed executing the code (in seconds). Furthermore, write the best result to a file named gm_tsp_n.txt where n is the value of n for that case. File format will be the same as in the first case.
3. Minimum Spanning Tree (MST) Method: This time you will try to find a solution by using MST method that is given as:
   i. Start with city 1.
   ii. Find the MST (details will be given in lab) using Prim's or Kruskal's algorithm.
   iii. Starting from root, traverse the spanning tree using depth first search (DFS).
   iv. Construct the path based on the order of the nodes visited in DFS.

   You will run this method for n values you tried at the first method and also for n=40, n=60, n=80, n=101. Report the results and time passed executing the code (in seconds). Furthermore, write the best result to a file named mst_tsp_n.txt where n is the value of n for that case. File format will be the same as in the first case.
4. 2OPT: In this case you will start with a valid random solution and use 2-OPT algorithm to find a better solution. The details of the 2-OPT algorithm will be given in Lab. The edges to be swapped

will be selected randomly. Your algorithm will stop if there is no improvement for the last 50 iterations. You will run this method for n values you tried at the first method and also for n=40, n=60, n=80, n=101. Report the results and time passed executing the code (in seconds). Furthermore, write the best result to a file named 2opt_tsp_n.txt where n is the value of n for that case. File format will be the same as in the first case.

5. 3OPT: In this case you will start with a valid random solution and use 3-OPT algorithm to find a better solution. The details of the 3-OPT algorithm will be given in Lab. The edges to be swapped will be selected randomly. At each step, you will select the tour that results in the shortest tour among the four possible alternatives. Your algorithm will stop if there is no improvement for the last 50 iterations. You will run this method for n values you tried at the first method and also for n=40, n=60, n=80, n=101. Report the results and time passed executing the code (in seconds). Furthermore, write the best result to a file named 3opt_tsp_n.txt where n is the value of n for that case. File format will be the same as in the first case.
6. Bonus: Any other non-trivial method will earn you bonus points. However, you can only implement one additional method.

What to deliver: You will submit your source code (.java files) and your report in a zip archive and mail them to cmpe160.projectsubmission@gmail.com with subject "CMPE160 Project3" (without quotation marks) until **1/06/2011, 17.00**. The mail address given is for project submissions, if you have questions use my regular e-mail address. Make sure that the name of the zip achieve is your student ID. Please write your name and number as comments to the top of each java file. Clear code with meaningful variable names and good commenting is requested. Your usage of javadoc will also be evaluated so try to use javadoc as much and as reasonable as possible. This time your report will be long. Besides details about your classes and your bonus method (if you implement one), it should give the results of the algorithms, comparison of the results with some charts.