Loel Nelson ICS 340

Report for Deliverable C

For deliverable C I decided to go with simulated annealing for 2 reasons; I saw somewhere that said it was a good answer to CSP and was good for larger files, also it was from a video on youtube that helped me out a lot in figuring out how to go about tackling the traveling salesman problem. I started by creating a class called Tour for handling my tour and grabbing the total distance of the tour. Next, I started working my algorithm for deliv C. The algorithm I chose was simulated annealing which models the physical process of heating and slowly lowering the temp to decrease the defects in metal (or other materials). I start my program by creating a stopwatch for checking how long the algorithm took from starting the program until the end when results were printed out to console and to file. Then, I went and checked the graph for unconnected nodes and if not connected I added “fake” edges of length 999,999. After that I started my algorithm of simulated annealing in the optimizeTour() method. I played around with the temp for the cooling of the “machine” which basically is how many iterations I will complete before the machine “cools” or stops, also I set the rate at which the machine cools coolingRate of type double, this really was the deciding factor in how long the code ran. Next, I made a tour from the starting node but then decided that instead I would just shuffle the tour right away, so I always started from a random tour. I printed the beginning results of the first tour grabbed and the total distance. I set the first tour as the best tour because at the time it is the best until further into while loop. I started the while loop to go through my tours and find the best one. Once in the loop I create a neighbor tour from the current tour and start the switching process. I grab a random position from the tour and use those 2 positions to switch the nodes. After switching the nodes in the neighbor solution, I grab the total distance from the current tour and the neighboring tour after switching nodes randomly. I send the neighbor tour and the current distance along with the temp to my acceptProb() method which is used to see if accepting the new tour as the best tour or not, all done with random double values, it returns either 1 for accepting the new tours distance or some random number below 1 if not. Now, I check if the current solution is the best or to keep going through and switching the nodes. If after 5 mins and still not a better tour I do a random restart of the tour. If, for some reason it does go longer I stop the loop at 15 mins if no better tours are found or if the temp is cooled and not any better tours in 20million iterations. I print each better tour and when finished with while loop I print the final solution and the elapsedTime() since starting Deliv C. The methods I used for the rest are just for manipulating the annealing and creating the stopwatch. I created two methods for finding random values used in the random walk and deciding to use the newest solution or not. I also have the method for creating the tour which is used to grab it from the start node but doesn’t matter because I randomize it soon after anyway. The last method I use is for checking the edges if connected or not for use in the fake edge method and if not connected returns a Boolean operator back to fakeEdge method to add the fake edge.

Ps. Bryce said his best was around 30 million for graph 0 and he said your best was 7k so I feel even though I don’t get the 2500 regularly it does happen every now and then so I felt really good about my implementation of the algorithm. This was a ‘fun’ project once I got the algorithm figured out and was able to play around with it. Hopefully by turning it in on time it will make up for missing the homeworks all the time. Noon is my nemesis. Also, if you run deliv C before running deliv A and B it can change the results because of C connecting the graphs