**AI Project 3**

Eric Dockery

Computer Engineering and Computer Science

Speed School of Engineering

University of Louisville, USA

[eadock01@louisville.edu](mailto:eadock01@louisville.edu)

Introduction:

For this assignment, we are asked to solve the traveling salesman problem using the closest edge insertion heuristic implementing a greedy algorithm. For this problem, we insert the vertex that is closest to an edge already in tour.

Approach:

The data sets given to test our code had an extra 6 lines that needed to be removed and stripped. My solution prompted the user for the tsp file then striped the file and sorted the information that was needed for the solution by storing that data in an array (list in python.) Once the data was sorted the program calls a function to find the closest two vertex’s in the problem. I chose the closest two vertex’s because it seemed more logical than Prim’s algorithm of randomly selecting a starting point. After the first two vertex’s are selected the program then finds the closest edge to that path and adds it to the tour. I originally had an if statement that was producing the incorrect path causing the traveling tour to have many crossing paths, but after sitting down and re-stepping through the paths and the logic I was able to correct this error. Although I used the closest points in the tour I believe that the best solution would result in finding the furthest points and calculating the original tour as an oval encompassing all of the possible node coordinates then modify this tour to collapse on itself and form the shortest tour by selecting the inner paths. I was not able to code this model in the time frame for the project.

After the tour is selected and calculated it is passed into a nested loop to plug the coordinates back into the vertex nodes. These values are then used and passed to a function that generates a GUI interface. I was having some trouble with the GUI being tiny due to the points being so close together. I tried to clean up the vertex’s by doing a light color modification, but it was overwriting the points so I just used a x2 magnification on the points and left the ovals clear. The GUI is not as clean as I would like, but I can improve it for future projects.

3. Results:

3.1 Data:

The data that was used for this assignment was generated using Concorde. The format for the data was:

NAME: concorde30

TYPE: TSP

COMMENT: Generated by CCutil\_writetsplib

COMMENT: Write called for by Concorde GUI

DIMENSION: 30

EDGE\_WEIGHT\_TYPE: EUC\_2D

NODE\_COORD\_SECTION

1 87.951292 2.658162

2 33.466597 66.682943

3 91.778314 53.807184

4 20.526749 47.633290

5 9.006012 81.185339

6 20.032350 2.761925

7 77.181310 31.922361

8 41.059603 32.578509

9 18.692587 97.015290

10 51.658681 33.808405

11 44.563128 47.541734

12 37.806330 50.599689

13 9.961241 20.337535

14 28.186895 70.415357

15 62.129582 6.183050

16 50.376904 42.796106

17 71.285134 43.671987

18 34.156316 49.113437

19 85.201575 71.837519

20 27.466659 1.394696

21 97.985778 44.746239

22 40.730003 98.400830

23 73.799860 61.076693

24 85.076449 17.029328

25 16.052736 11.899167

26 20.160527 67.238380

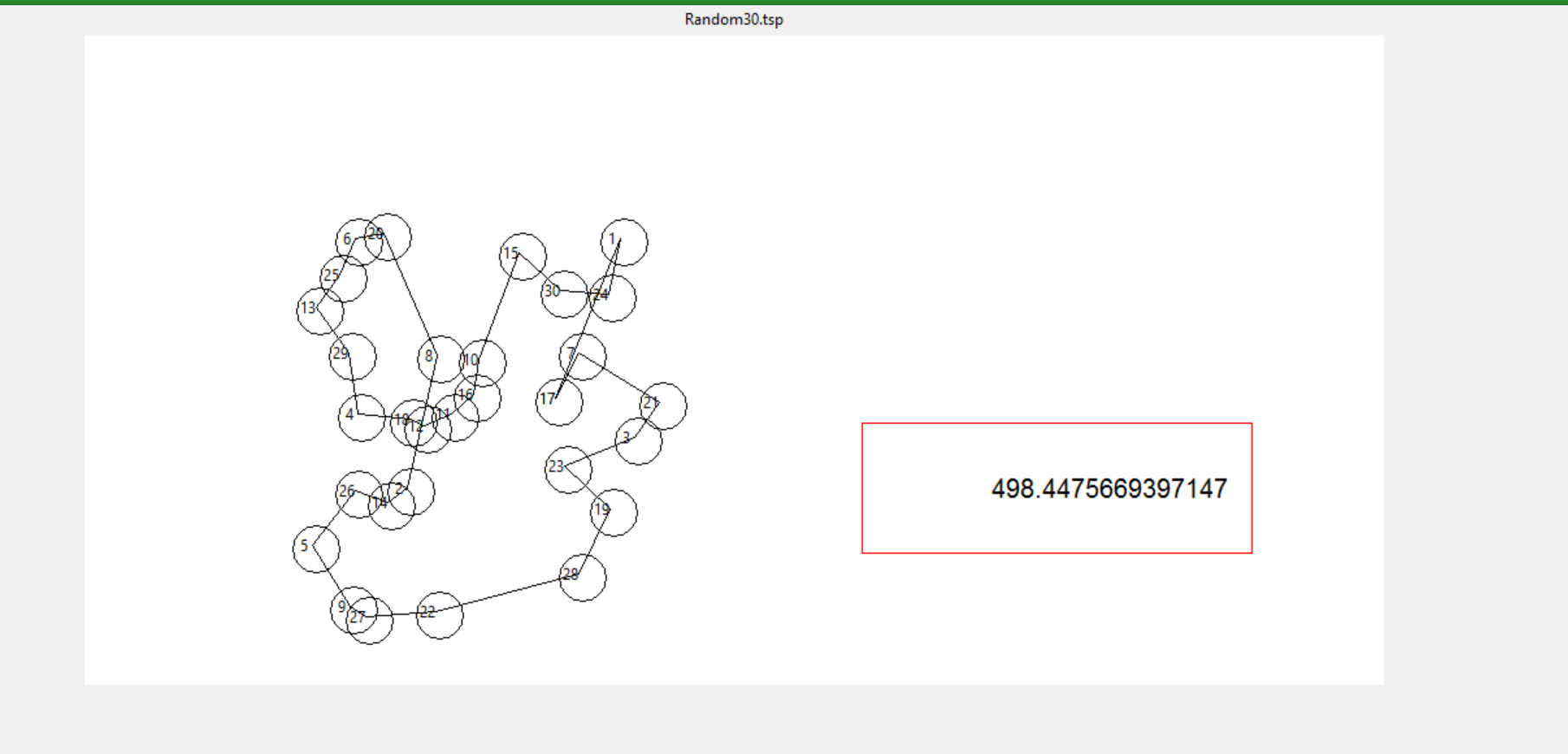
27 22.730186 99.725333

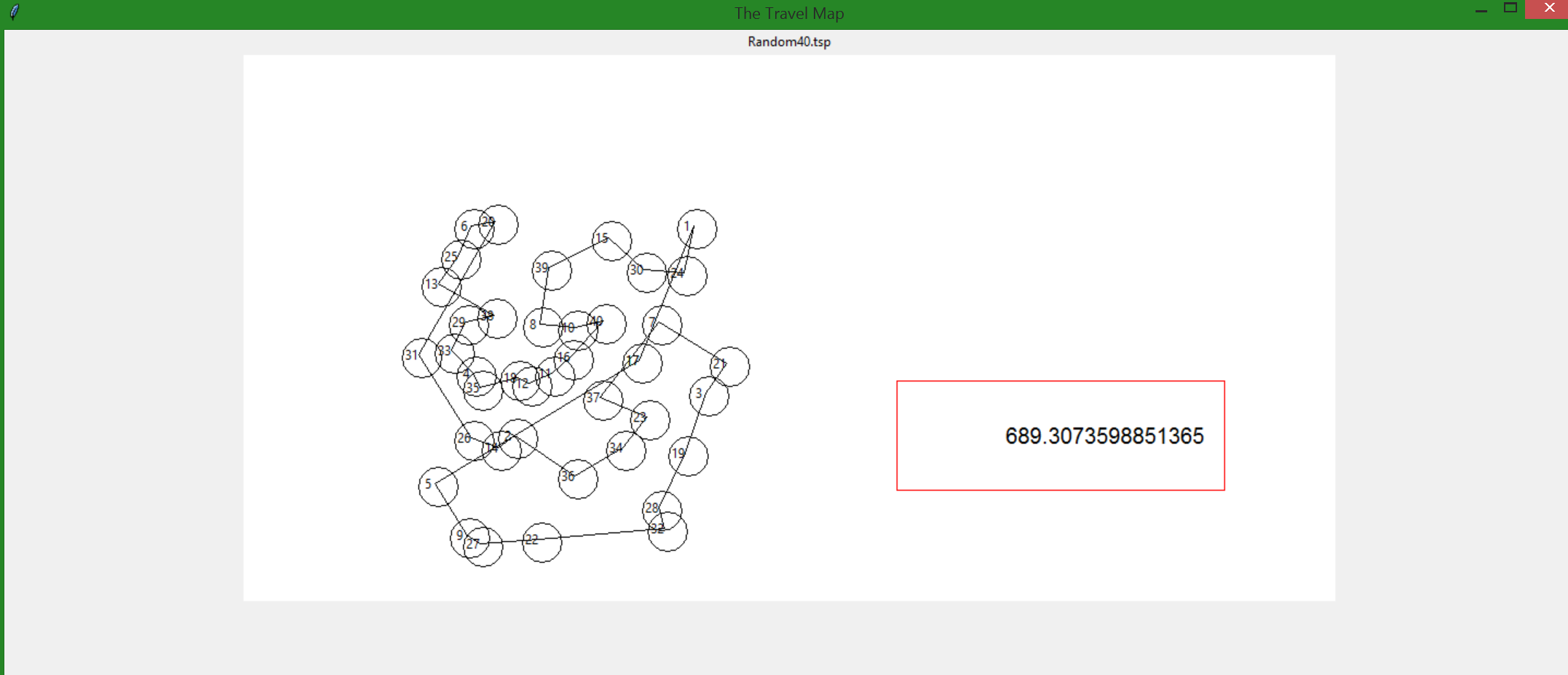
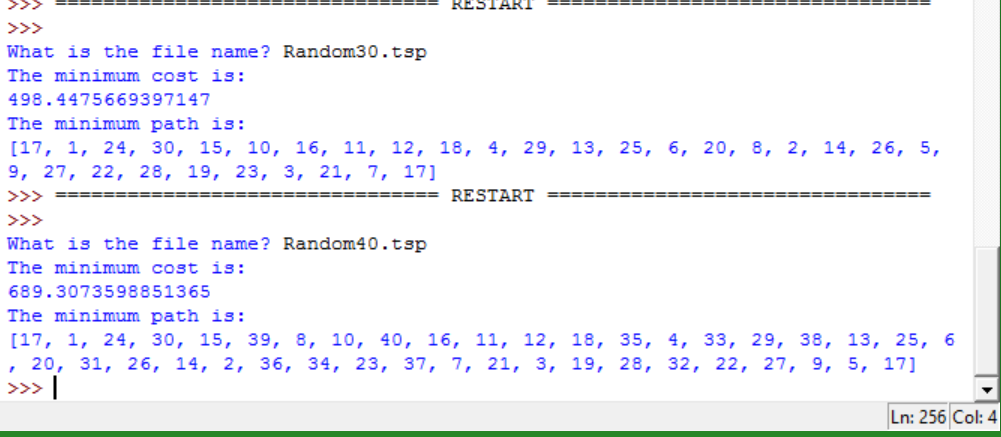
28 77.196570 88.503677

29 18.494217 31.971191

30 72.743919 16.071047

3.2 Results:





4. Discussion:

.My program has a few intersecting lines, but the run time is very efficient. I don’t believe that my results are 100% correct but I do think that it is close to the optimal solution. I think that if the program took the furthest two nods and made an oval path encompassing the whole dataset, then modifying that path to collapse the oval to the nodes, then a better result would be generated. Unfortunately I was unable to define a logical response to this theory in the time frame for this project.

5. References:

<http://web.tuke.sk/fei-cit/butka/hop/ConstructiveHeuristicsForTheTSP.pdf>