Project 4 TSP

Joel Smith
James Guerra
Mark Dillman

Project Report

Description of three methods/algorithms for solving the traveling salesman problem:

One method for solving the traveling salesman problem is the greedy approximation approach. Due to the problem being NP Complete, a greedy algorithm selects vertices to visit based on some greedy criteria. Without thinking ahead, the algorithm visits every vertex.

A second method for solving the traveling salesman problem is using a MST. A algorithm for finding a MST is found, such as Kruskal's algorithm, for intance. The program then travels along that MST visiting every vertex.

A third method for solving the traveling salesman problem is using brute force. All possible combinations of routes are examined, and the smallest routes are selected. This approach is not very resource efficient.

Verbal description of algorithms used in this project:

This project uses a greedy algorithm to approximate a traveling salesman route. The shortest path to another vertex is chosen from a current vertex, so long as that vertex has not already been visited. The program traverses the path until all vertices have been visited. The program then returns to the beginning vertex.

Discussion of why this algorithm was selected:

This algorithm was selected in order to compute a path quickly. In order to compete in the time competition, it was decided that a greedy algorithm would provide good running time

Pseudo Code:

```
read file, insert into vector all vertices
compute distance between all vertices
choose a starting vertex
while all havent been visited:
    visit the closest one that hasnt been visited
    mark the visited vertex as visited
    repeat
```

Best Tours for the three example instances:

example1:

.00433 seconds

170766/170742 = 1.00014056

example2:

.2333861 seconds

2890 / 2866 = 1.00837404

example3:

259376 seconds

112321988 / 112314640 = 1.00006542334997

Best Tours for the competition instances:

test-input-1.txt:

0.001928 seconds

27865 / 27842 = 1.00083

test-input-2.txt:

0.01267 seconds

48721 / 48680 = 1.00084

test-input-3.txt:

0.167318 seconds

123892 / 123774 = 1.00095

test-input-4.txt:

1.27076 seconds

247639 / 247392 = 1.001

test-input-5.txt:

    10.2512 seconds

    522667 / 522217 = 1.00086

test-input-6.txt:

    84.2454 seconds

    1050413 / 1049477 = 1.00089

test-input-7.txt:

    1209.84 seconds

    2579701 / 2577388 = 1.0009