Bryan Cook
Brian Maxey
CS 3510

Traveling Salesman Problem

Problem:
From a starting city, travel to every city in a graph only visiting each city once, then return to the starting city.

Objective:
Find a minimum Hamiltonian cycle with the most efficient running time.

Background before our solution:
This is an NP-hard problem because there is no way to solve it in polynomial time. The brute force approach of checking every possible path of cities is O(n!) but this is far too inefficient to realistically do at any large scale. There is a known deterministic dynamic programming solution using the Held-Karp algorithm that gets this down to $O(2^n n^2)$ but this is still far too inefficient for our purposes. Because the problem presented to us is more about speed of calculation and not the truly optimal minimum distance path, we decided to choose a heuristic approach that may sacrifice optimality of the traveling cost for overall computing speed.

Our Approach:
We decided to take on this problem using the simulated annealing heuristic based off of the cooling of metal. We start at a high temperature, then slowly reduce by multiplying it by a fixed alpha that is slightly less than one during each iteration of trying to find a lower cost route. The route starts off random and goes through random changes with more extreme changes at higher temperatures. As the cost and temperature lowers, it will go through more but less extreme random changes until settling at a stable global minimum temperature. The temperature reduces in O(logn) time. For each temperature reduction, O(n) searches for attempted and accepted changes occur. If a new tour is rejected, it takes O(1) time but if it is accepted, the reversal takes O(n) city exchanges. Bringing the running time of this heuristic approach to be $O((n^2+n)logn))$.Since most changes take place at low temperatures, O(nlogn) is not negligible compared to $O(n^2 logn)$. Our algorithm also has to read all the files from the text document which is in O(n) time and pre-calculates distances between every city which in $O(n^2)$time. These operations are negligible compared to the simulated annealing aspect- the precalculating distances actually reduces the time since it makes all costs calculations O(1) time during the annealing instead of O(n).

Pseudocode for Annealing:

```
Let tour = start_node
For i = 0, i < imax:
      T = temperature (i⁄imax)
      Pick a random adjacent node,  new_tour = neighbor(tour[i])
      If cost(new_tour) < cost(tour) and
            exp((cost(new_tour) - cost(tour)) / T) > random(0, 1)
                  Then: tour = new_tour
Output: tour
```

Our 10 Runs:

```
Total Cost: 28918
Tour:
16 20 21 23 18 17 14 13 9 7 3 4 8 5 1 2 6 10 11 12 15 19 22 29 28 26 25 27 24 16
Total Cost: 28593
Tour:
22 21 17 14 13 9 7 3 4 8 5 1 2 6 10 11 12 15 19 18 20 16 24 27 25 26 28 29 23 22
Total Cost: 28084
Tour:
4 5 1 2 6 11 10 8 12 15 19 18 17 21 22 23 29 28 26 20 25 27 24 16 14 13 9 7 3 4
Total Cost: 27750
Tour:
22 23 21 29 28 26 25 27 24 16 20 17 14 13 9 7 3 4 8 5 1 2 6 10 11 12 15 19 18 22
Total Cost: 27603
Tour:
14 13 9 7 3 4 8 5 1 2 6 10 11 12 15 19 18 17 21 22 23 29 28 26 20 25 27 24 16 14
Total Cost: 27750
Tour:
1 2 6 10 11 12 15 19 18 22 23 21 29 28 26 25 27 24 16 20 17 14 13 9 7 3 4 8 5 1
Total Cost: 27603
Tour:
18 19 15 12 11 10 6 2 1 5 8 4 3 7 9 13 14 16 24 27 25 20 26 28 29 23 22 21 17 18
Total Cost: 27603
Tour:
25 27 24 16 14 13 9 7 3 4 8 5 1 2 6 10 11 12 15 19 18 17 21 22 23 29 28 26 20 25
Total Cost: 27750
Tour:
11 10 6 2 1 5 8 4 3 7 9 13 14 17 20 16 24 27 25 26 28 29 21 23 22 18 19 15 12 11
Total Cost: 27750
Tour:
16 24 27 25 26 28 29 21 23 22 18 19 15 12 11 10 6 2 1 5 8 4 3 7 9 13 14 17 20 16
```

Average Total Cost: 27940.4
Standard Deviation: 458.1