Traveling salesman Problem

(Mandatory Assignment 1 INF3490)

Thomas Kaspar Parmer

September 25, 2015

1 Assignment

In this assignment i have solved the traveling salesman problem. The goal of the problem is to find a tour through a set of cities or positions, which starts at the first city, visits each city exactly once and returns to the first city, such that the path traveled is as short as possible. This problem is known to be NP-complete , i.e. no serial algorithm exists that runs in time polynomial in n, only in time exponential in n, and it is widely believed that no polynomial time algorithm exists. Because of this we often want to compute an approximate solution when the number of cities increases.

2 Exhaustive search

The following algorithm generates all possible solutions, and picks the shortest one. This algorithm searches all $(n-1)!$ possible paths, compares them, and keeps the one with the shortest distance. The shortest path of the first 10 cities: 5272. The exhaustive search took 50 seconds to find it. To find shortest path through all 24 would take 50s (time of 10 cities) times 11 times 12 times 13... up to 24. This equals $24!/10! * 50$ wich translates to around 271085341 millenniums. By this number it is easy to see that the exhaustive search becomes quickly becomes unusable when number of cities are increased.

3 Hill climbing

The hillclimb algorithm performs very well and is my favorite of the algorithms. As can be seen in the code (file "hill.py") hillclimbing is a quite simple algorithm that can produce good and often optimal results. It is though, not without drawbacks as it can find a suboptimal sulotion and believe its the best one, finding a local top. Here are my results for 10 and 24 cities.

Number of cities: 10 Time used: 0.2010s

Best tour: 5272 Longest tour: 7149 Mean: 6400 Standard deviation: 581

Number of cities: 24 Time used: 8.2286s

Best tour: 11806 Longest tour: 16020 Mean: 13727 Standard deviation: 851

4 Genetic algorithm

The genetic algorithm is able to find a solution in a very short time. Although it might not find the best solution, if you run it enough times you can find a decent solution, even on larger problems. I have used to different methods to solve the traveling salesman problem with a genetic algorithm. The algorithm starts with a random tour every time it is run. then, given a percentage chance, creates another offspring and does a partial mapped crossover where the parents are combined to a new child, and/or a mutation where parts of the route is swapped. Hopefully, this tour will be better the parent. If so, the child becomes the new parent. New children tours are repeatedly created until the desired number of generations is reached. Here are my results for 24, 10 and 6 cities.

Number of cities: 24 Time used: 0.0892s

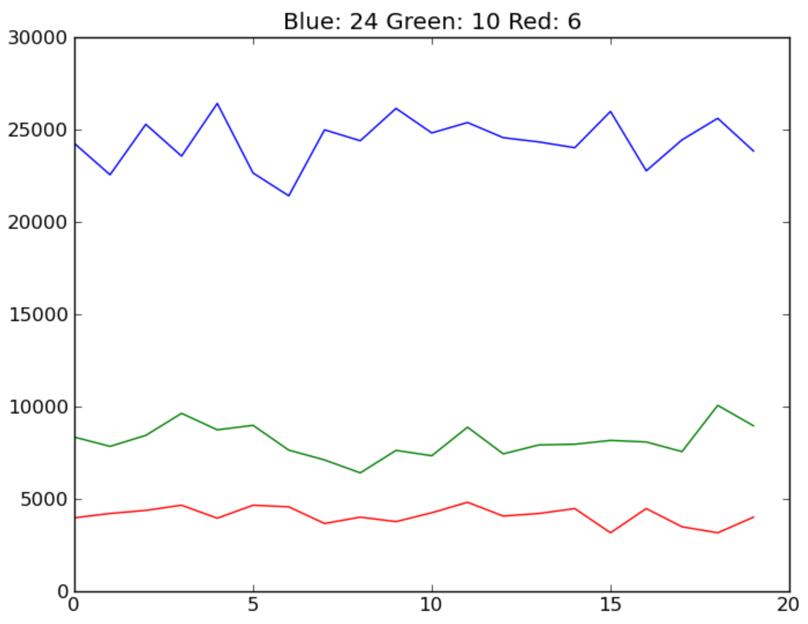
Best tour: 21412 Longest tour: 26656 Mean: 24370 Standard deviation: 1277

Number of cities: 10 Time used: 0.1051s

Best tour: 6411 Longest tour: 12311 Mean: 8158 Standard deviation: 853

Number of cities: 6 Time used: 0.1172s

Best tour: 3167 Longest tour: 6451 Mean: 4100 Standard deviation: 460



The ga did not find the shortest tour on 10 cities, but was not so far off. In running time it was very fast, as it only checks a fraction of the solutions compared to the exhaustive search witch checks everything. I would say my implementation of the ga algorithm is only worth using when the quite precise hill climb starts to become too slow.