INF3490/INF4490 Mandatory Assignment 1: Travelling Salesman Problem

Student: Alexander Wiig Sørensen
User name: alexawii

Exhaustive Search

Exhaustive search was attempted with incrementally increasing the number of cities. Starting at 6 cities until the computation time was deemed to big.

6 cities
Best route: [0, 1, 4, 5, 2, 3]
Barcelona - Belgrade - Bucharest - Budapest - Berlin - Brussels
Best distance: 5018.80999
Computation time: 5.189 ms

10 cities
Best route: [6, 8, 3, 7, 0, 1, 9, 4, 5, 2]
{Copenhagen - Hamburg - Brussels - Dublin - Barcelona - Belgrade - Istanbul - Bucharest - Budapest - Berlin}
Best distance: 7486.3099
Computation time: 38987.524 ms

Based on the exponential increase in computation time, I would expect an exhaustive search with all cities to take more than 70 000 years.

Hill Climbing

My hill climbing algorithm first generated a random permutation of the selected range of cities (10 initially, and then 24). It iteratively swapped integers in the permutation and always kept the the result if it was better than the last. These are the results from 20 runs:

10 cities
{Best route: [9, 1, 5, 2, 6, 8, 3, 7, 0, 4]
Istanbul - Belgrade - Budapest - Berlin - Copenhagen - Hamburg - Brussels - Dublin - Barcelona - Bucharest}
Best distance: 7603.24
Worst distance: 11278.62
Average distance: 10022.8715
Standard deviation: 820.983465645
Computation time: 34.812 ms

24 cities
Best route: [8, 16, 11, 19, 14, 0, 12, 6, 4, 20, 17, 22, 9, 18, 1, 3, 13, 7, 21, 10, 23, 15, 5, 2]
{Hamburg - Paris - London - Saint Petersburg - Moscow - Barcelona - Madrid - Copenhagen - Bucharest - Sofia - Prague - Vienna - Istanbul - Rome - Belgrade - Brussels - Milan - Dublin - Stockholm - Kiev - Warsaw - Munich - Budapest - Berlin}
Best distance: 25328.94
Worst distance: 29043.640000000007
Average distance: 27111.2665
Standard deviation: 1100.87288342
Computation time: 454.518 ms

Genetic Algorithm

I tried many different values. The genetic algorithm I ended with uses chooses parents based on a tournament setup, with 10 percent of the population being eligible for selection. It uses a swap mutation that has a 25 percent chance of triggering on each individual. The child is created by using an mpx algorithm. In my tests I was able to reach to optimal route with only a population size of 100. The running time was larger than expected, depending on the population size of course. With a size of 10, the time was around 0.883 ms, with a size of 100 the time was around 38.281 ms, and with a size of 1000 the time was around 3093.447 ms.

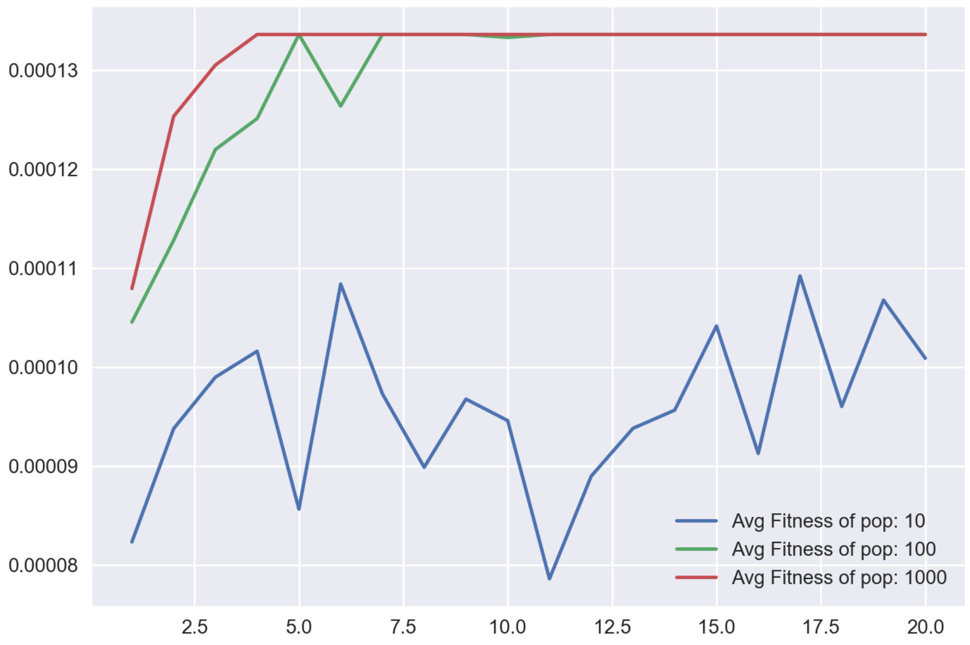
Population 10
Best distance: 9159.08
Worst distance: 12725.880000000001
Average distance: 10525.5985
Standard deviation: 916.010646774
[4, 9, 2, 0, 3, 7, 8, 6, 1, 5]
Bucharest - Istanbul - Berlin - Barcelona - Brussels - Dublin - Hamburg - Copenhagen - Belgrade - Budapest

Population 100
Best distance: 7486.31
Worst distance: 10237.923
Average distance: 7608.943
Standard deviation: 542.01118
[1, 9, 4, 5, 2, 6, 8, 3, 7, 0]
Belgrade - Istanbul - Bucharest - Budapest - Berlin - Copenhagen - Hamburg - Brussels - Dublin - Barcelona

Population 1000
Best distance: 7486.31
Worst distance: 9267.43
Average distance: 7608.943
Standard deviation: 396.697824308
[1, 9, 4, 5, 2, 6, 8, 3, 7, 0]
Belgrade - Istanbul - Bucharest - Budapest - Berlin - Copenhagen - Hamburg - Brussels - Dublin - Barcelona

Hybrid Algorithm

The hybrid algorithm is an augmentation of the genetic algorithm. The hill climb function has been split up to easily perform a hill climb operation on a simple list on integers. This is done on each child prior to adding them to the population. I tried both Lamarckian and Baldwin approach, to add fitness with and without altering the permutation itself. Through my tests I experienced that keeping both the result and the fitness level would decrease the performance of my algorithm. If I instead kept only the fitness level and kept the child unaltered, the results were much better.

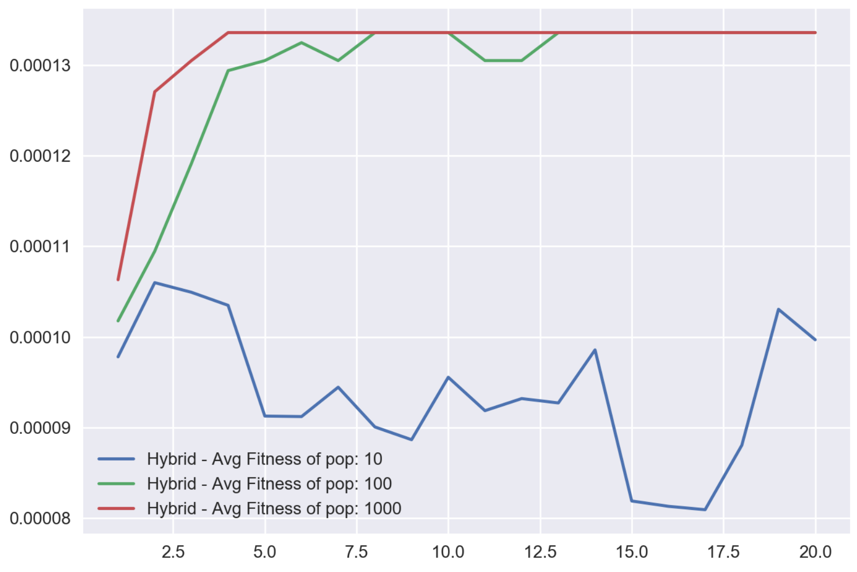


Maybe some of my parameters were off. I would appreciate if the method to see the benefit could be suggested in the review of my assignment, as I would really like to understand this. With only the fitness level affected by the hill climb, the results I got were generally very comparable to those of the genetic algorithm. As the results were worse if I also included the new permutation, I fail to see the immediate benefit of this variant, though that could very well be because of how I have implemented it. I would also love some comments on this.

Population 10
Best distance: 9159.08
Worst distance: 12725.880000000001
Average distance: 10525.5985
Standard deviation: 916.010646774
[4, 9, 2, 0, 3, 7, 8, 6, 1, 5]
Bucharest - Istanbul - Berlin - Barcelona - Brussels - Dublin - Hamburg - Copenhagen - Belgrade - Budapest

Population 100
Best distance: 7486.31
Worst distance: 10237.923
Average distance: 7608.943
Standard deviation: 542.01118
[1, 9, 4, 5, 2, 6, 8, 3, 7, 0]
Belgrade - Istanbul - Bucharest - Budapest - Berlin - Copenhagen - Hamburg - Brussels - Dublin - Barcelona

Population 1000
Best distance: 7486.31
Worst distance: 9267.43
Average distance: 7608.943
Standard deviation: 396.697824308
[1, 9, 4, 5, 2, 6, 8, 3, 7, 0]
Belgrade - Istanbul - Bucharest - Budapest - Berlin - Copenhagen - Hamburg - Brussels - Dublin - Barcelona



How to run the code

The file ‘tsp.py' is ready to be run as it is. All that is needed is uncomment whatever block you want to run. In the case of the hybrid algorithm, you need to change the function name ‘hybrid_evolve_population’ to ‘evolve_population’ where it is called to change between the hybrid and normal genetic algorithm. The Lamarckian and Baldwin settings are found in ‘hybrid_evolve_population’, and can be commented in/out as desired.