

Inf3490

Mandatory Assignment 1

Marius Madsen

Mariumh

To run it. Simply type: python ob.py

Exhaustive search:

observing the amount of time used when incrementing with one, you will notice that it takes time for $n-1$ cities smaller times n . this would mean $n!$

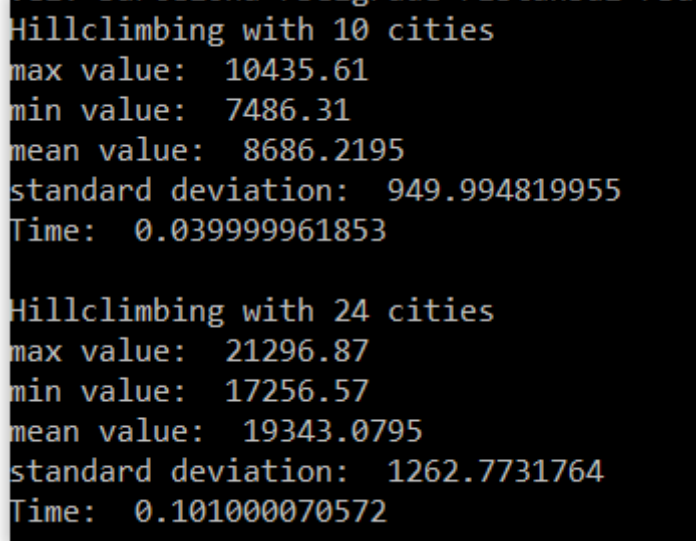
The best route with 10 cities is 7486.31 km long

With the tour Barcelona->Belgrade->Istanbul->Bucharest->Budapest->Berlin->Copenhagen->Hamburg->Brussels->Dublin->Barcelona
the time used is approximately 3.24 seconds
running for 24 cities would be $3.24 * 24! / 10!$

Hillclimbing:

Hillclimbing with ten cities gives the same best result but uses 0.04 seconds which is a speedup of 81 times

Example run of hillclimbing with statistics:



```
Hillclimbing with 10 cities
max value: 10435.61
min value: 7486.31
mean value: 8686.2195
standard deviation: 949.994819955
Time: 0.039999961853

Hillclimbing with 24 cities
max value: 21296.87
min value: 17256.57
mean value: 19343.0795
standard deviation: 1262.7731764
Time: 0.101000070572
```

Genetic Algorithm:

I chose to use inversion mutation, where it chooses a random length of the permutation to inverse.

Also I chose Pmx since it's a good crossover algorithm when working with permutations

I chose populations 25 50 and 100

the parameters are how many elites, how many mutant and crossover offsprings. Every parameter is supposed to be choosed as a percentage. If mutant+crossover >1 or is equal or less than zero it will result in a crash. having 0 offspring doesn't make sense either also more than 1 would result in population increasing, which I assumed is not what we want for this instance I chose to have 10 percent elite and a quarter of each offspring

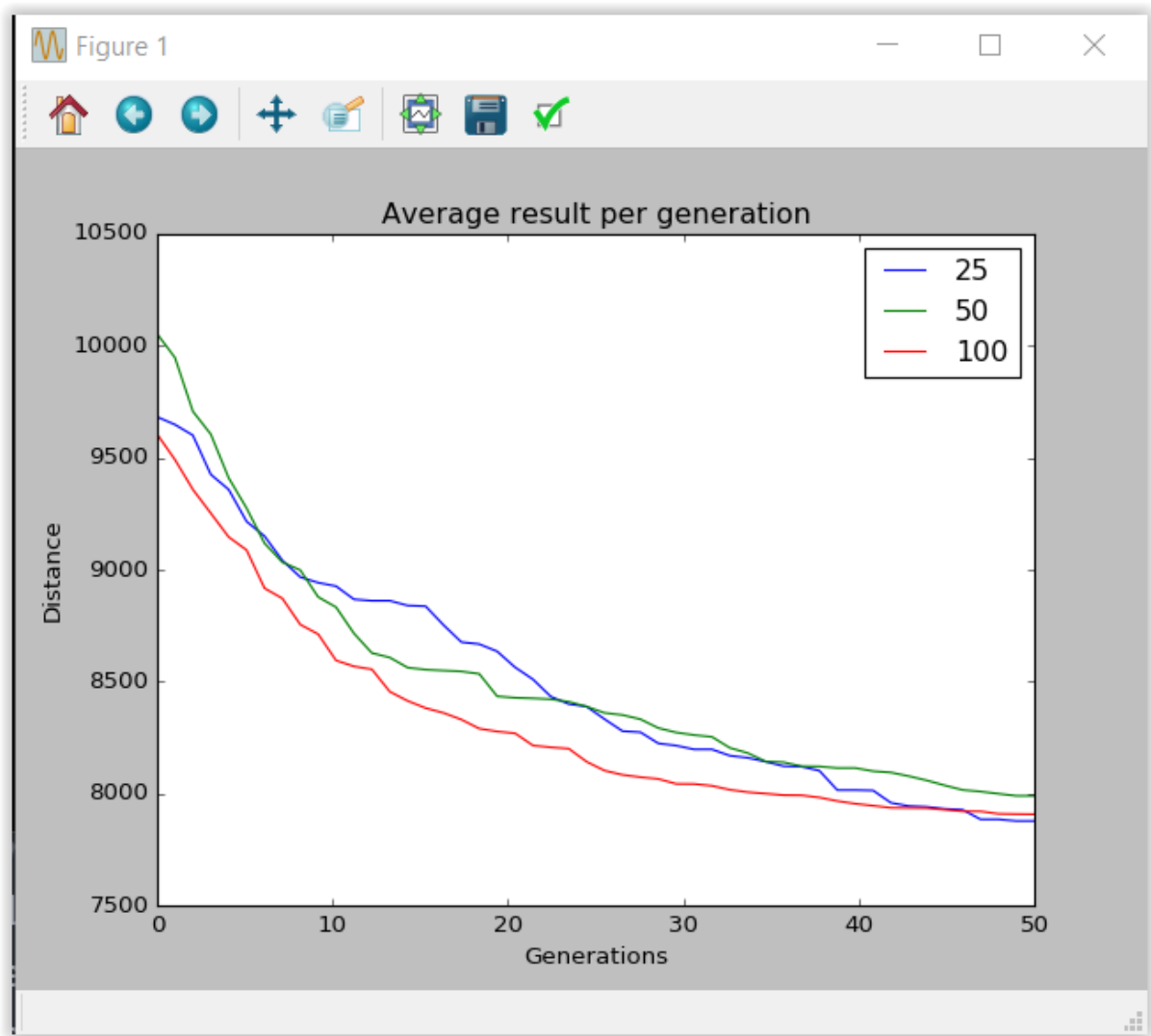
Ran with both 10 cities and 24 cities:

10 cities:

```
Genetic Algorithm with 10 cities and a population of 25
max value: 9681.5905
min value: 7879.4965
mean value: 8502.04947
standard deviation: 518.338719269
Time: 0.304000139236

Genetic Algorithm with 10 cities and a population of 50
max value: 10052.008525
min value: 7991.773825
mean value: 8514.8317835
standard deviation: 521.551066507
Time: 1.18499994278

Genetic Algorithm with 10 cities and a population of 100
max value: 9604.53392625
min value: 7909.49519125
mean value: 8322.48039917
standard deviation: 461.108856577
Time: 4.632999897
```



Cities = 24:

```
Genetic Algorithm with 24 cities and a population of 25
max value: 27176.2831963
min value: 21635.6507596
mean value: 23729.71877
standard deviation: 1635.45962319
Time: 0.643000125885

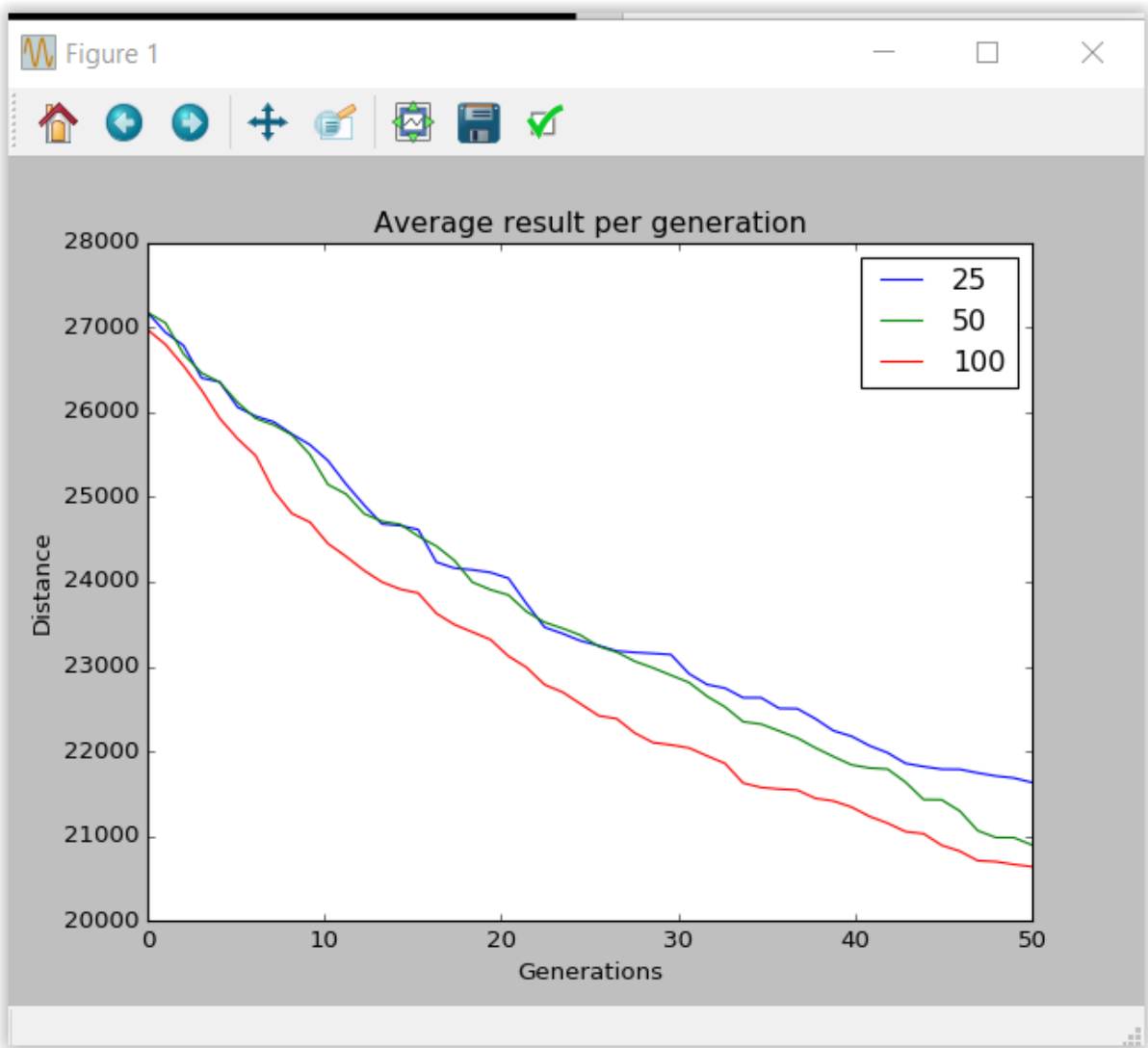
Genetic Algorithm with 24 cities and a population of 50
max value: 27174.6861598
min value: 20898.733538
mean value: 23555.1415185
standard deviation: 1792.19550748
Time: 2.60700011253

Genetic Algorithm with 24 cities and a population of 100
max value: 26973.818308
min value: 20643.1351769
mean value: 22948.5222059
standard deviation: 1815.15886834
Time: 10.0439999104
```

You can tell that for more generations it will get lower, but you can also see that its starting flatten out also, but looking at the sums, you'll notice that they didn't reach their global minimum

For the 10 first cities it almost reaches the shortest route, it has the possibility of reaching it

For 10 cities it is almost the same as exhaustive search (3-4 second) but for 24 cities, exhaustive search would never do it in 10 seconds



What I've done is that I've made a populations with different permutations of the trip, then I have let x offspring be made by mutation and y offspring be made by crossover. To select which parent that is making a kid, I've chose to use rank selection. I also did that to choose parents to fill in next population. There is also a small group of elites made, that secures the best answer (about 10 percent of population)

