INF3490 Mandatory Assignment 1

Johannes Kjernlie (johannkk)

September 2016

1 Instructions

All the source code for the assignemnt can be found at GitHub: `https://github.com/Kjernlie/INF3490/tree/master/oblig1`. It is also included in the delivery on devilry.

To run the programs just type in *python chosenprogram.py* in the terminal. The variables in the programs that can be changed in order to run for different number of cities and population etc. should hopefully be easily recognizable in the source code files.

I also want to mention that I collaborated with Øyvind Huuse on parts of the exericse.

2 Exhaustive search

For the exhaustive search algorithm I created a help function which returned the total distance of a tour given the tour route and the distance matrix. For finding all permutations I used the permutations function from itertools. I then go trough all permutations and finds the shortest tour.

This method will always give the correct result, but at high computational cost. For 10 cities it checks $10! = 3628800$ combinations and uses 44.5 seconds. If we want to use it with all of the 24 cities, it would have to check $24! = 6.204484e + 23$ combinations. We can expect that the computational time will be around $\frac{6.204484e+23}{3628800} = 1.7097895e + 17$ times higher than for 10 city case, which is unfeasible on personal computers.

The terminal output of the code is given below

```
"""
The elapsed time for N = 6  is  0.00620198249817  seconds.
The elapsed time for N = 7  is  0.0489280223846  seconds.
The elapsed time for N = 8  is  0.402410984039  seconds.
The elapsed time for N = 9  is  3.91781592369  seconds.
The elapsed time for N = 10  is  44.4848008156  seconds.

 This is the shortest tour distance  7486.31
```

```
The best travel route is as follows ['Barcelona', 'Dublin', '
    Brussels', 'Hamburg', 'Copenhagen', 'Berlin', 'Budapest',
    'Bucharest', 'Istanbul', 'Belgrade']
"""
```

3 Hill Climbing

We can see that the hill climbing algorithm doesn't yield the correct result, when we compared with the exhaustive search algorithm, altough the results is not too far off. However, when the execution time of the program is paramount, the hill-climbing method will be the better choice. For 10 cities our hill-climbing algorithm used 0.05 seconds, while the exhaustive search used 44.5 seconds. Note that the hill-climbing yields slightly different results every time, but if you make the algorithm do a high number of iterations, you would get a reasonable good result each time. This would require a larger computational time, but it would still be small, compared to the exhaustive search algorithm. Also, we can see that from the terminal output box underneath that the execution times for the 24 city case is only slightly higher than for the 10 city case.

```
# For 10 cities
"""
Elapsed time for  10  cities, is  0.0475599765778  seconds.
The best tour is  7503.1
The worst tour is  10180.55
The mean of the tours is  8922.212
The standard deviaton of the tours is  741.616037971
"""

# For 24 cities
"""
Elapsed time for  24  cities, is  0.0690491199493  seconds.
The best tour is  21784.42
The worst tour is  27723.41
The mean of the tours is  25650.3525
The standard deviaton of the tours is  1471.6979365
"""
```

4 Genetic Algorithm

In this genetic algorithm I decided to use partially mapped crossover and inversion mutation, for the crossover and mutation operators, respectively. I think these operators will be a good choice for the traveling salesman problem since they will keep some of the adjacency of the lists, which will be important here.

In the program I set up tournament groups with 5 tours in each group. The number of groups will depend on the size of the population, and is given by

$$\text{Number of tournaments} = \frac{\text{Population size}}{\text{Tournament size}}.$$

Note that I have made sure that the number of tournaments always will be an integer that goes up in the population size.

In each tournament group I find the two best tours, and use those as the two parents. From the two parents I get a child, which I mutate with a probability of 3%. Additionally, I also find the worst tour in each tournament.

When creating the new population I delete the worst tours from each tournament and insert the new children. This will make the populations size remain the same size for every generation, while at the same time contain better and better tours.

The terminal outputs with the best result, worst result, mean and standard deviation is given in the box underneath. In Fig. (1) and (2) the average tour length for 10 and 24 cities with various populations sizes is plotted over 150 generations. When we compare the results for 10 city case with the exhaustive search algorithm, we can see that our genetic algorithm also yields the lowest possible tour for population sizes of 100 and 150. Additionally, for 24 cities we obtain a much better result than the hill-climber algorithm does. All of this leads us to believe that our genetic algorithm is a good choice for solving the traveling salesman problem. It both gives good results and is computationally efficient.

```
# Terminal output
# For a population of 50 and  10 cities
"""
The best tour is  7791.86  with population size  50
The worst tour is  11867.341  with population size  50
The mean of the tours is  8044.692224  with population size
    50
The standard deviaton of the tours is  749.934471998  with
    population size  50
"""

# For a population of 100 and  10 cities
"""
The best tour is  7486.31  with population size  100
The worst tour is  12213.2522  with population size  100
The mean of the tours is  7808.15731933  with population size
    100
The standard deviaton of the tours is  899.006471627  with
    population size  100
"""

# For a population of 150 and  10 cities
"""
```

```
The best tour is  7486.31  with population size  150
The worst tour is  12129.9804667  with population size  150
The mean of the tours is  7840.38100356  with population size
    150
The standard deviaton of the tours is  871.253094014  with
    population size  150
"""

# For a population of 50 and 24 cities
"""
The best tour is  18733.29  with population size  50
The worst tour is  30841.667  with population size  50
The mean of the tours is  21193.6299507  with population size
    50
The standard deviaton of the tours is  2794.14361598  with
    population size  50
"""

# For a population of 100 and 24 cities
"""
The best tour is  18128.48  with population size  100
The worst tour is  30562.6222  with population size  100
The mean of the tours is  20304.82253  with population size
    100
The standard deviaton of the tours is  2755.96455571  with
    population size  100
"""

# For a population of 150 and 24 cities
"""
The best tour is  17606.39  with population size  150
The worst tour is  30633.8865333  with population size  150
The mean of the tours is  19643.4409827  with population size
    150
The standard deviaton of the tours is  2948.51425662  with
    population size  150
"""
```

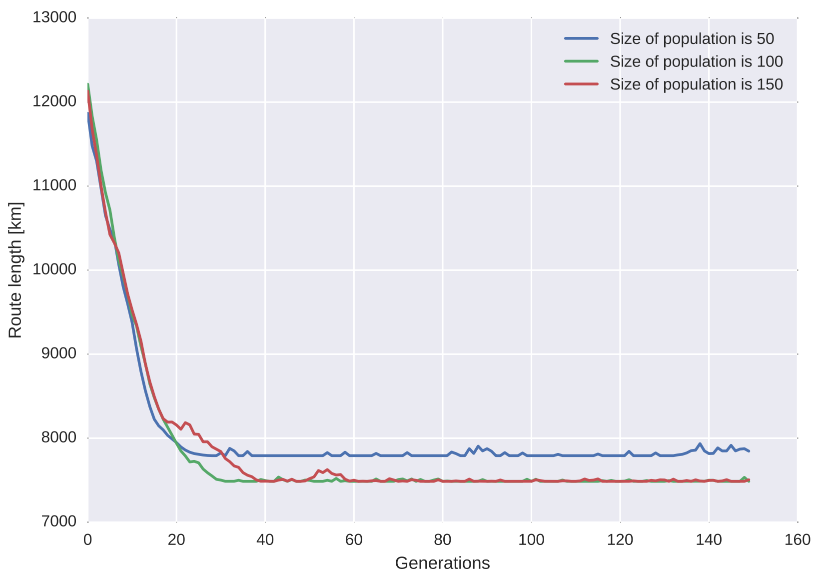

Figure 1: The average tour length for 10 cities in the traveling salesman problem is plotted for a population of 50, 100 and 150.

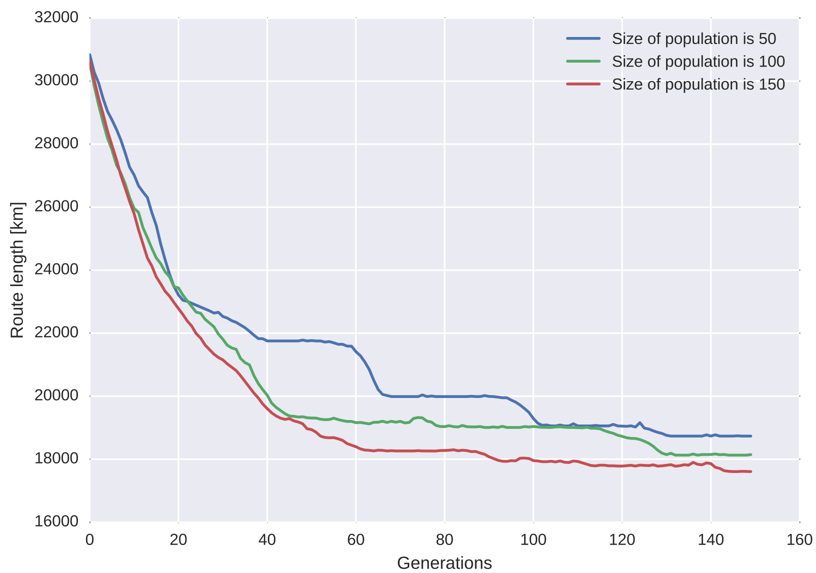

Figure 2: The average tour length between 24 cities in the traveling salesman problem is plotted for a population of 50, 100 and 150.