INF3490 Mandatory Assignment 1: Travelling Salesman Problem

Name: Øyvind Huuse
Username: oyvinhuu

Instructions on how to run the program

The program is written in Python.

The code is divided into four programs: TSP_data.py, exhaustive.py, hillclimber.py and genetic.py.

To run the exhaustive.py program you need to enter the number of cities to include, into the terminal window (python exhaustive.py <number of cities>).

To run the hillclimber.py program you need to enter the number of cities to include and how many iterations to complete into the terminal window (python hillclimber.py <number of cities> <iterations>). Also, the number of runs can be changed in the code (line 23).

To run the genetic.py program you need to enter the number of cities to include, the population size and how many generations to go through into the terminal window (python genetic.py <number of cities> <population size> <number of generations>). Also, the tournament size, number of runs and probability for mutation can be changed in the code (line 27-29).

Exhaustive search

The shortest trip for 10 cities was: 7486.310 km
shortest path =
['Copenhagen', 'Hamburg', 'Brussels', 'Dublin', 'Barcelona', 'Belgrade', 'Istanbul', 'Bucharest', 'Budapest', 'Berlin']

and the time it took was:
time = 201.53 seconds

For 24 cities it would take 24! = 6.204484*10^23 runs to cover every tour. The 10 cities tour, took 10! = 3628800 and that demanded 201.53 seconds. This is 3628800/6.204484*10^23 = 5.8486733*10^-18 of the 24 city tour. So it would possibly take about 201.53s/5.8486733*10^-18 = 3.44573871 × 10^{19} seconds (/(60s*60min*24t*365d)) = 34 624.3033 years.

Hillclimber

Hill climber starts with a random route and changes two cities with each other for 1000 iterations. The best route is kept at the end of each iteration.

The shortest tour the hillclimber found was: 7680.49. This is only 194.18 longer than the shortest trip (7486.31).
The shortest trip for 10 cities was (at a random try): 7845.94
['Brussels', 'Dublin', 'Barcelona', 'Istanbul', 'Bucharest', 'Budapest', 'Belgrade', 'Berlin', 'Copenhagen', 'Hamburg']
and the time it took was:
time = 0.06 seconds.

Compared to Exhaustive search, Hill Climbing uses 0.029% of the time and at this particular run, the trip was 359.63 km longer for the "Hill Climbing" than for the "Exhaustive Search".

20 runs for 10 cities:

Best trip	
7729.01	
8243.97	
8265.27	
8015.15	
7680.49	BEST TOTAL
7915.34	
8301.74	
8267.25	
8005.72	
8204.06	
7830.01	
8724.02	
8744.36	WORST TOTAL
8366.01	
8601.19	
7887.03	
8608.6	
8472.38	
8699.06	
8050.72	

Best tour	7680.49
Worst tour	8744.36
Mean	8230.57
Standard deviation	328.16
time	1.21

20 runs for 24 cities:

Best trip

25104.55	
23796.91	
25249.45	
24094.27	
24454.12	
25466.89	WORST TOTAL
23459.73	
23134.38	
24747.47	
25216.19	
25182.23	
25123.96	
24727.31	
23901.95	
24149.49	
23631.02	
23894.82	
22978.72	
22303.91	BEST TOTAL
22894.83	

Best tour	22303.91
Worst tour	25466.89
Mean	24175.61
Standard deviation	898.94
time	2.70

Genetic Algorithm

For the crossover I have used “partially mapped crossover”. This crossover method was used to minimize the number of cities that lost contact with its neighbor city and thereby keep the good route intact. But, still introducing some renewal by crossing routes from two parents.

For the mutation I have used “inversion” of a random segment size of the children. Inversion is used in order to not introduce too much randomness in the evolution of the code. The probability for mutation is tested at 0.3. This is to ensure new route alternatives appear after many generations of selection.

The program was tested with three population sizes: 10, 30 and 90 individuals. The tournament size used was 10 and the program ran for 100 generations each time. 20 runs were completed each time to get more valuable result concerning data

Figure 1 shows how the performance improve on average over the 20 runs in each generation for 10, 30 and 90 individuals in the population for 10 cities.

20 runs for 10 cities, 10 in population:

Best trip	
10972.01	
8843.22	
9053.89	
7969.09	
8979.76	
10428.27	
8963.78	
10363.96	
9388.58	
8005.72	
10086.62	
9922.84	
7663.51	BEST TRIP
8516.81	
7745.8	
8974.42	
10001.01	
11244.37	WORST TRIP
8602.52	
9567.28	

Best tour	7663.51
Worst tour	11244.37
Mean	9264.67
Standard deviation	1018.93
time	3.44

20 runs for 10 cities, 30 in population:

Best trip

7663.51	
7952.3	
8557.48	
7817.42	
7845.94	
8587.57	
7680.49	
8717.52	
8663.73	
8398.26	
8577.48	
7680.49	
9085.61	WORST TRIP
7737.95	
8033.93	
7486.31	BEST TRIP
8827.24	
8199.77	
8204.06	
8459.23	

Best tour	7486.31
Worst tour	9085.61
Mean	8208.81
Standard deviation	452.61
time	9.60

20 runs for 10 cities, 90 in population:

Best trip

7603.24	
7503.1	
7737.95	
7486.31	
7737.95	
8265.17	
7486.31	
7486.31	
8407.18	
7503.1	
7663.7	
7486.31	BEST TRIP
8204.06	
7486.31	
7737.95	
7486.31	BEST TRIP
7486.31	BEST TRIP
8436.39	WORST TRIP
7969.09	
7887.69	

Best tour	7486.31
Worst tour	8436.39
Mean	7753.037
Standard deviation	322.45
time	28.21

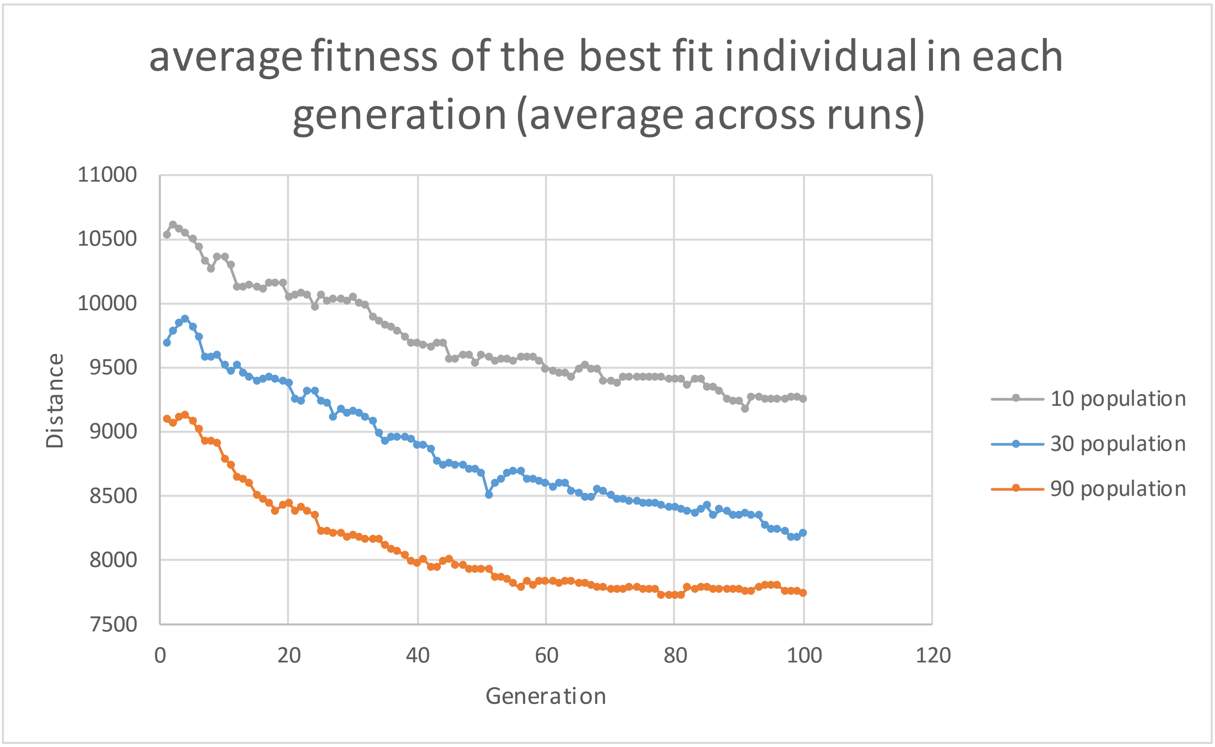

Figure 1

Discussion

It seems that the greater the population size, the better average fit in each generation (across the runs). The fit is better, even though all other parameters are kept constant (number of generations, mutation probability, number of cities, tournament size and number of runs). Looking at the best trip over 20 runs, the results are close for all three population sizes (7486.31-7663.51). The variation between the worst trip is a little bigger (8436.39-11244.37), with the best results for the biggest population either way and the worst results for the smallest population. The standard deviation drops drastically from 10-30 individuals in the population, and is still dropping some between the 90- and 30 population. The running time is growing steadily with a tripling as the population size triples. Figure 1 also demonstrates how larger populations in general gives better results. It shows that for the 90 population, it would maybe be enough with 60 generations on average to get a decent result, as the curve starts to flatten out here. For the 30 population the curve is stall decreasing steadily at the 100th generation. The 10 population is possibly starting to flatten out towards the end, suggesting that one tournament with all the individuals inside might get to a final result earlier than the 30 population with three tournaments. But, the results for the 10 population is not close to as good as the 30 (or 90) by far.

Regarding evolution time, even for 20 runs in the GA, the 90 population only takes 28 seconds. This is about 14% of the running time of the exhaustive search. Still, it seems like the hillclimber might be the best choice for this problem. Using only 4.3% of the computational time as the 90 population, it gets an average route only 477 longer than GA 90 population. Their standard deviation is almost the same.

Conclusion

A large population gives better results than a small one. The standard deviation is close for the two largest populations. This could suggest that further increase in population size would not better the standard deviation very much (but it might still better the shortest trip results). More generations could help the 10 population test to reach the shortest route, but is not necessary for the larger populations. In fact, the 90 population finds the shortest tour three times during the generation, suggesting that it could possibly due with some fewer generation and still get the job done. Figure 1 shows that the shortest tour possibly is discovered already after about 60 generations, which is close to half of what was used. Still, there is no guaranty that the best route would be found here. On average it doesn't find it after 100 generations even. This can suggest that some parameters should be adjusted. Possibly, it is better to have relative tournament sizes closer to what we can see in the 30 population.

The GA can run for more generations to better the probability of getting the optimal route and still use a lot less computer time than the exhaustive search. But, hillclimber is much less time consuming than both of them. With more running time it will possibly outperform the GA. Although, a fine-tuning of the variables in the GA might increase its quality as well.

Further work

It would be interesting to see how the change of generation and tournament size will influence the results for the 90 population in GA. Fewer and bigger tournaments would increase the competition and thereby possibly better the results, but we would have fewer new children in each generation which means more generation would be demanded to get a similar or larger introduction of new tour alternatives. The effect of mutation could also be tested out by increasing the probability. With fewer tournaments an increase in mutation probability could play an important role in keeping the population varied.

Extra questions:

"Among the first 10 cities, did your GA find the shortest tour (as found by the exhaustive search)? Did it come close? For both 10 and 24 cities: How did the running time of your GA compare to that of the exhaustive search? How many tours were inspected by your GA as compared to by the exhaustive search? "

The GA found the shortest tour (7486.31) for both the 90- and the 30-population, three times in the 90 population.

For the ten first cities with 90 individuals in the population, GA found (for one particular run through):

```
 To compare with the exhaustive search, the GA found the shortest trip:  7486.31
 citylist for the last generation =
 ' Bucharest', 'Budapest', 'Berlin', 'Copenhagen', 'Hamburg', 'Brussels', 'Dublin', 'Barcelona',
 'Belgrade', 'Istanbul'
time =  1.38 seconds
```

This is the same route as in the exhaustive search, only that the start city is shifted three times to the right. The computational time is only 0.8% of the exhaustive search with the GA.

For all the 24 cities with 90 individuals in the population, GA found (for one particular run-through):

```
the shortest length for 24 cities (last generation):  17492.0
citylist for the last generation =
['Munich', 'Barcelona', 'Madrid', 'Paris', 'London', 'Dublin', 'Rome', 'Brussels', 'Milan',
'Budapest', 'Vienna', 'Prague', 'Berlin', 'Stockholm', 'Saint Petersburg', 'Moscow', 'Kiev',
'Warsaw', 'Bucharest', 'Istanbul', 'Belgrade', 'Sofia', 'Hamburg', 'Copenhagen']
time =  2.65 seconds
```

In the GA for 10 and 24 cities, ran through the 90 population 100 times (number of generations), if you don't count the 20 times the code was repeated to get better data. This gives 90*100= 9000 tours roughly (some 100 extra with children being created, possibly mutation and adding information to lists). The exhaustive run through covered 10!= 3628800 tours. The GA did 0.23% of the tours that the exhaustive program did. This explains why the GA demands just a fraction of the computational time the exhaustive search does.

No run through was done with exhaustive search for 24 cities, but we can see from the 10 city exhaustive search that it would take drastically longer than for the GA.