IN3050 - Assignment 1: Traveling Salesman Problem

Name: kristian Gyene, Username: kristsgy

Exhaustive search

How to run the script: python3 exhaustive_search.py

The program will ask the user how many cities to be included. A subset of the cities from the european_cities.csv will be used, starting with Barcelona.

For exhaustive search I used itertools.permutations to get all possible combinations of routes. Then it goes through every single route and calculates the shortest possible one.

What is the shortest tour (i.e., the actual sequence of cities, and its length) among the first 10 cities (that is, the cities starting with B,C,D,H and I)? How long did your program take to find it?

Result from running the script with 10 cities:
Shortest tour: ['Copenhagen', 'Hamburg', 'Brussels', 'Dublin', 'Barcelona', 'Belgrade', 'Istanbul', 'Bucharest', 'Budapest', 'Berlin', 'Copenhagen']
Tour distance: 7486.310km
Time result: 26.232s

Calculate an approximation of how long it would take to perform exhaustive search on all 24 cities?

For 10 cities, there is 10! = 3628800 possible combinations to compute and this took 26.232s.
For 24 cities, there is 24! = 6.204 × 10^23 possible combinations to compute.

26.232 seconds = $8{,}3 * 10^{-7}$

$$\frac{3628800}{8{,}3 * 10^{-7}} = \frac{6.204 * 10^{23}}{x}$$

$$x = 141901455026 = 1{,}4 * 10^{11}$$

Exhaustive search for 24 cities will take a lot of years. My calculation ended up with $1{,}4 * 10^{11}\ years$

Hill climbing

How to run the script: python3 hill_climbing.py

The program will ask the user how many cities to be included and how many times the algorithm should run. A subset of the cities from the european_cities.csv will be used, starting with Barcelona. I first started to generate all permutations and pick a random one, but then I found out that it was to heavy with 24 cities. So I chose to use random.sample to get only one permutation. For finding neighbors I used random.sample to swap two random cities in the tour.

I just followed the instructions in lecture 2 for the algorithm:
• Pick a random tour as the current best
• Compare to neighbor tours
• If the neighbor is shorter, replace the current best (The neighbor = another random tour)
• Repeat until we reach a certain number of evaluations (I chose 1000)

How well does the hill climber perform, compared to the result from the exhaustive search for the first 10 cities?
The hill climber is much faster because it doesn't have to go through all possible solutions, instead it find a solution that it's happy with. When it comes to accuracy, the hill climber isn't that accurate. It really depends on how many starting point and how many neighbors it checks, and that will cost time.

Report the length of the tour of the best, worst and mean of 20 runs (with random starting tours), as well as the standard deviation of the runs, both with the 10 first cities, and with all 24 cities.

20 runs – 10 cities:
----------BEST RESULTS----------
Shortest tour: ['Hamburg', 'Brussels', 'Dublin', 'Barcelona', 'Belgrade', 'Istanbul', 'Bucharest', 'Budapest', 'Berlin', 'Copenhagen']
Tour distance: 7486.310000km

----------WORST RESULTS----------
Longest tour: ['Barcelona', 'Dublin', 'Copenhagen', 'Berlin', 'Budapest', 'Belgrade', 'Istanbul', 'Bucharest', 'Hamburg', 'Brussels']
Tour distance: 8419.090000km

Mean distance: 7944.674444km
standard deviation: 431.359183km

Time result: 3.891937s

20 runs – 24 cities:
----------BEST RESULTS----------
Shortest tour: ['Warsaw', 'Prague', 'Munich', 'Milan', 'Rome', 'Barcelona', 'Madrid', 'Dublin', 'London', 'Paris', 'Brussels', 'Hamburg', 'Berlin', 'Copenhagen', 'Stockholm', 'Saint Petersburg', 'Moscow', 'Kiev', 'Bucharest', 'Istanbul', 'Sofia', 'Belgrade', 'Budapest', 'Vienna']
Tour distance: 12325.930000km

----------WORST RESULTS----------
Longest tour: ['Hamburg', 'Dublin', 'Madrid', 'Barcelona', 'Rome', 'Sofia', 'Belgrade', 'Budapest', 'Vienna', 'Munich', 'Prague', 'Berlin', 'Warsaw', 'Bucharest', 'Istanbul', 'Milan', 'Paris', 'London', 'Brussels', 'Kiev', 'Moscow', 'Saint Petersburg', 'Stockholm', 'Copenhagen']
Tour distance: 16177.670000km

Mean distance: 14131.918500km
standard deviation: 982.286178km
Time result: 5.473760s

Genetic algorithm

How to run the script: python3 genetic_algorithm.py

The program will ask the user how many cities to be included. A subset of the cities from the european_cities.csv will be used, starting with Barcelona.

The flow chart is a representation of the algorithm I implemented. First I selected a random population by using random.shuffle on a route. Then for every population I started to select parents based on their fitness. The children are the products from a crossover (Partially Mapped Crossover), and mutation (swap-mutation). After getting twice as many individuals because of the offspring, I just removed half of the population wich was also the weakest considering fitness. For population size I used 30, 60 and 90. When the number of generations hits a curtain number of generation without improvement (100), then it’s done with the algorithm.

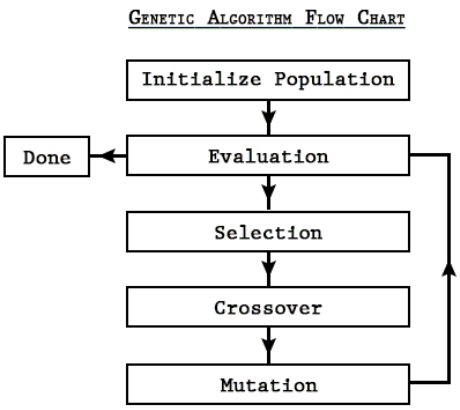


Figure 2

Population 30

Run	1	2	3	4	5	6	7	8	9	10
Distance	15363.45	14340.84	13841.93	14436.98	13526.80	14947.75	14998.06	13426.44	14749.61	15430.12
Run	11	12	13	14	15	16	17	18	19	20
Distance	14873.75	13627.48	12745.68	14470.48	12999.36	13694.65	13639.53	14226.59	13868.85	14634.36

Best distance: 12745.68km
Worst distance: 15430.12km
Mean distance: 14192.14 km
Standard deviation: 757.40km

Population 60

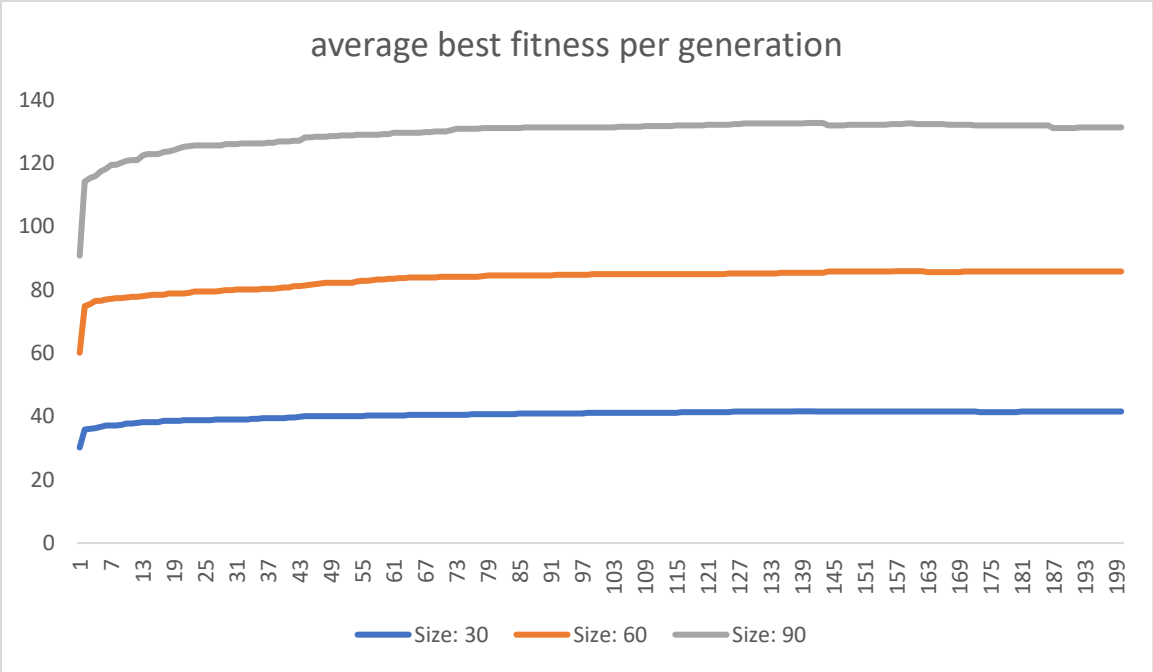
Run	1	2	3	4	5	6	7	8	9	10
Distance	12628.99	13165.64	14009.57	13969.34	12842.74	13545.90	14277.78	13397.09	13268.71	13532.98
Run	11	12	13	14	15	16	17	18	19	20
Distance	15136.94	13476.76	13295.44	13009.93	14966.01	13594.18	13878.29	14018.88	12514.02	13761.01

Best distance: 12514.02km
Worst distance: 15136.94km
Mean distance: 13614.51km
Standard deviation: 679.21km

Population 90

Run	1	2	3	4	5	6	7	8	9	10
Distance	12808.92	13020.34	13473.65	12805.99	12340.49	12287.07	13002.71	13360.80	12769.46	13274.89
Run	11	12	13	14	15	16	17	18	19	20
Distance	12325.93	13647.19	12799.99	12927.65	13000.90	12703.47	12734.14	12558.67	12402.42	12423.08

Best distance: 12287.07km
Worst distance: 13647.19km
Mean distance: 12833.39km
Standard deviation: 391.19km



Note: the data for this diagram is also included in the delivery

Every single route got a fitness that is calculated by route_distance / total_distance_in_population. The runs stopped at different generations so I chose to only use the first 200 as the generation number for each of the variants. I also used the generation when the algorithm terminated, not the generation where it found the best result.

Conclude which is best in terms of tour length and number of generations of evolution time.

It really doesn't seem like the population size is that important when we look at the graphs, but the problem is that it's not consistent enough. If accuracy is the most important thing, then the population size 90 is the best. For running time, size 30 is clearly the best, because it does not have to go through a lot of generations.

Among the first 10 cities, did your GA find the shortest tour (as found by the exhaustive search)? Did it come close?

10 cities:
------ POPULATION SIZE: 30 ------
Number_of_generations: 137
Best individual stats:
route: ['Berlin', 'Budapest', 'Bucharest', 'Istanbul', 'Belgrade', 'Barcelona', 'Dublin', 'Brussels', 'Hamburg', 'Copenhagen']
Distance: 7486.3099999999995
Fitness: 43.649530543795606
Mean distance: 7897.084091km

standard deviation: 361.183254km
Average: 7905.405185185186

------ POPULATION SIZE: 60 ------
Number_of_generations: 42
Best individual stats:
route: ['Dublin', 'Brussels', 'Hamburg', 'Copenhagen', 'Berlin', 'Budapest', 'Bucharest', 'Istanbul', 'Belgrade', 'Barcelona']
Distance: 7486.3099999999995
Fitness: 89.3617099950654
Mean distance: 7905.405185km
standard deviation: 415.526418km
Average: 7846.320487804877

------ POPULATION SIZE: 90 ------
Number_of_generations: 62
Best individual stats:
route: ['Hamburg', 'Brussels', 'Dublin', 'Barcelona', 'Belgrade', 'Istanbul', 'Bucharest', 'Budapest', 'Berlin', 'Copenhagen']
Distance: 7486.3099999999995
Fitness: 134.14213814196222
Mean distance: 7846.320488km
standard deviation: 287.051828km

24 cities:
------ POPULATION SIZE: 30 ------
Number_of_generations: 268
Best individual stats:
route: ['Berlin', 'Hamburg', 'Copenhagen', 'Warsaw', 'Kiev', 'Moscow', 'Saint Petersburg', 'Stockholm', 'Dublin', 'Madrid', 'Barcelona', 'London', 'Brussels', 'Paris', 'Budapest', 'Bucharest', 'Istanbul', 'Sofia', 'Belgrade', 'Rome', 'Milan', 'Munich', 'Vienna', 'Prague']
Distance: 14873.75
Fitness: 40.98597780256544
Worst distance: 22410.660000000003
Mean distance: 16287.704000km
standard deviation: 1566.690911km

------ POPULATION SIZE: 60 ------
Number_of_generations: 151
Best individual stats:
route: ['Madrid', 'Barcelona', 'Rome', 'Sofia', 'Istanbul', 'Bucharest', 'Berlin', 'Hamburg', 'Dublin', 'London', 'Brussels', 'Warsaw', 'Copenhagen', 'Stockholm', 'Saint Petersburg', 'Moscow', 'Kiev', 'Belgrade', 'Budapest', 'Vienna', 'Prague', 'Munich', 'Milan', 'Paris']
Distance: 15136.949999999999
Fitness: 83.27361220169603
Worst distance: 21840.87
Mean distance: 16967.035000km
standard deviation: 1559.046462km

------ POPULATION SIZE: 90 ------
Number_of_generations: 296
Best individual stats:
route: ['Rome', 'Milan', 'Munich', 'Prague', 'Warsaw', 'Vienna', 'Budapest', 'Belgrade', 'Sofia', 'Istanbul', 'Bucharest', 'Kiev', 'Moscow', 'Saint Petersburg', 'Stockholm', 'Copenhagen', 'Berlin', 'Hamburg', 'Brussels', 'Paris', 'London', 'Dublin', 'Madrid', 'Barcelona']
Distance: 12325.930000000002
Fitness: 135.19240578385802
Worst distance: 18069.54
Mean distance: 13687.674865km
standard deviation: 1432.048263km
Time result: 32.598450s

The GA finds the best tour among 10 cities just like exhaustive search, but there is a big difference in efficiency. It also found a great solution on 24 cities.

For both 10 and 24 cities: How did the running time of your GA compare to that of the exhaustive search?
It was a lot faster, and also found the shortest tour for 10 cities. However, it could not find the absolute best solution for 24 cities, but found pretty decent solutions. By the time the exhaustive search is finished running after 26.232s with 10 cities, the GA have already calculated three different solutions with three different population sizes in around 11.0 – 16.0s. That's a huge difference. The GA running time really depends on how big the population size is, and how many generations it goes through before terminating.

How many tours were inspected by your GA as compared to by the exhaustive search?
I ran the genetic algorithm on 24 cities with population size of 30. The particular run ran for 250 generations, assuming the algorithm does not generate duplicated children, this means about 2500 to 3000 tours. The exhaustive search would had to evaluate 6.204×10^{23} tours.