IN3050/IN4050 Mandatory Assignment 1

Traveling Salesperson Problem

Martin Mihle Nygaard (`martimn`)

February 2022

I've put all algorithms in a separate Python file, so see this for actual implementations. I'll just comment on my choices and the results here. I apologize if it's a bit messy, but had some issues with Jupyter Notebook `:(`

Libraries used: `numpy`, `matplotlib`, `pandas`, `seaborn`, and `basemap`.

Exhaustive Search

See plan plots in figure 1.

As we can see from the timing data in figure 2, the number of cycles involved in a exhaustive search grows *fast*. Time per cycle is roughly 1.1×10^{-5}, and further extrapolation gives:

$$1.1 \times 10^{-5} \text{ s} \cdot (24-1)! \approx 2.84 \times 10^{17} \text{s} \approx 9.01 \times 10^{9} \text{ years}$$

Hill Climbing

See plots in figure 3. Descriptive statistics in figure 4.

Genetic Algorithm

I've followed the book closely in my implementations. Starting with the skeleton sketched in the book (Eiben and Smith 2015, 26), I've tried to fill inn with appropriate operators.

- Parent selection: *Stochastic universal selection* (Eiben and Smith 2015, 84).
- Recombination: *Order crossover* (Eiben and Smith 2015, 73), since we want to preserve some structure, and order matters.
- Mutation: *Inversion mutation* (Eiben and Smith 2015, 69), again to preserve structure and not completely upend ordering.

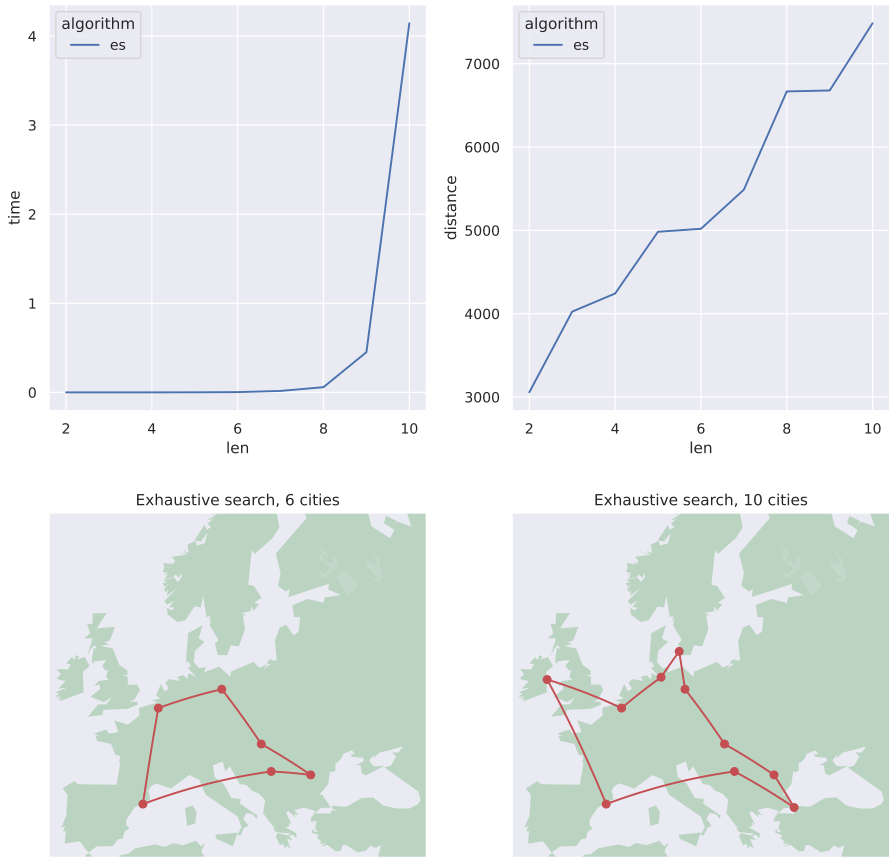

Figure 1: Results from exhaustive search algorithm

Time	Plan	Distance	Length	Cycles
0.000111	(0, 1)	3056.26	2	1
0.000113	(0, 1, 2)	4024.99	3	2
0.000289	(0, 1, 2, 3)	4241.89	4	6
0.000858	(0, 3, 2, 4, 1)	4983.38	5	24
0.001266	(0, 1, 4, 5, 2, 3)	5018.81	6	120
0.010637	(0, 3, 6, 2, 5, 4, 1)	5487.89	7	720
0.057756	(0, 1, 4, 5, 2, 6, 3, 7)	6667.49	8	5040
0.44933	(0, 1, 4, 5, 2, 6, 8, 3, 7)	6678.55	9	40320
4.149134	(0, 1, 9, 4, 5, 2, 6, 8, 3, 7)	7486.31	10	362880

Figure 2: Timing data from running exhaustive search algorithm

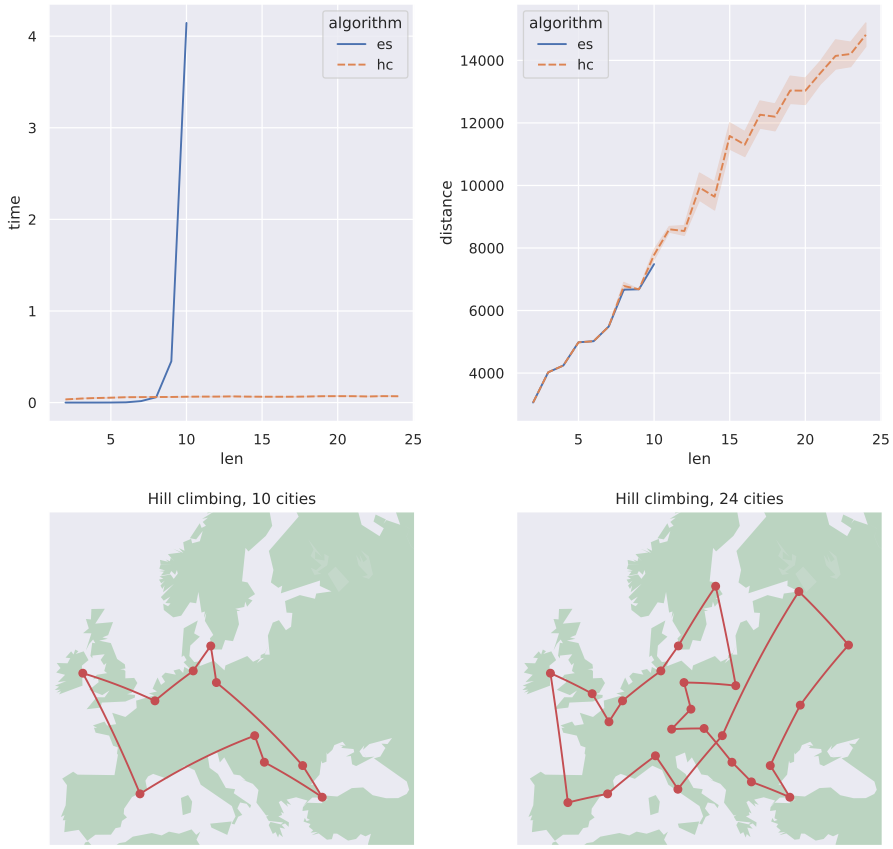

Figure 3: Hill climbing compared to exhaustive search. Plots of best plans.

	10 cities	24 cities
count	20	20
mean	7724.352000	14973.347500
std	359.785261	1568.472048
min	7486.310000	12924.480000
25%	7486.310000	14072.477500
50%	7503.100000	14832.480000
75%	7737.950000	15693.680000
max	8407.180000	19815.290000

Figure 4: Descriptive statistics of 20 example runs of hill climbing algorithm with 10 and 24 cities.

- Evaluation: Sorting the population using individual plan distance.
- Survivor selection: Rank based. Keeping it simple, using the sorted population, select the λ best. The book warns against premature convergence, though. I went for $(\lambda + \mu)$, choosing from both parents and offspring, which might have been a mistake. The book outlines reasons for a (λ, μ) preference, though most don't apply in this case. (Eiben and Smith 2015, 88–90)

The parameters … *Sigh.* The book suggests a λ/μ ratio of 5–7 if you want a high selection pressure (Eiben and Smith 2015, 89). I'm unsure. I tried to do *a lot* of random runs to see if I could find some signal, but found mostly noise. I've included a JSON file with data from over 72000 runs with random parameters (half with a pure genetic algorithm, i.e not hybrid). I've included a correlation matrix in figure 5. It *seems* like increasing λ and lowering μ has a positive effect on distance, but adverse effect on time. Same with generations. Size is not too important, but this is likely due to me disallowing $\mu <$ size. But my dataset is flawed, I think, because the constraints I've set on the random parameters (i.e. the intervals they're chosen from), is pretty arbitrary. I've also included some statistics from the 100 best solutions in figure 6, and would choose parameters based on those, but these statistics does not give any insurance on the robustness or reliability of these parameters.

	Time	Dist.	λ	μ	s	Gen.s	Size	λ/μ
Time	1.0000	0.1105	0.1306	-0.1623	0.0777	0.1238	0.0779	0.5563
Dist.	0.1105	1.0000	-0.0797	0.0675	-0.0294	-0.1505	0.0016	-0.0675
λ	0.1306	-0.0797	1.0000	0.2989	0.0091	-0.0407	0.0824	0.2523
μ	-0.1623	0.0675	0.2989	1.0000	0.0257	-0.0458	0.3178	-0.5285
s	0.0777	-0.0294	0.0091	0.0257	1.0000	-0.0127	0.0153	0.0135
Gen.s	0.1238	-0.1505	-0.0407	-0.0458	-0.0127	1.0000	0.0100	-0.0203
Size	0.0779	0.0016	0.0824	0.3178	0.0153	0.0100	1.0000	-0.1620
λ/μ	0.5563	-0.0675	0.2523	-0.5285	0.0135	-0.0203	-0.1620	1.0000

Figure 5: Correlation matrix for parameters, based on around 37000 runs with random parameters.

After 20 (new) runs using parameters inspired by figure 6, gives the statistics in figure 8, for plans of length 24 with population sizes 10, 50, 100. I've plotted time and distances in figure 9. There doesn't really seem to be much difference in distance, surprisingly, but calculation time increases with size. A sample generational plot is shown in figure 7. Finally, the best tours found in 20 runs are shown in figure 10.

	Time	Dist.	λ	μ	s	Gen.s	Size	λ/μ
mean	0.77	12357.46	42.84	14.37	1.53	67.93	63.08	5.84
std	0.94	42.73	12.71	12.38	0.17	20.61	21.84	5.43
min	0.17	12287.07	5.00	2.00	1.21	19.00	16.00	0.92
25%	0.39	12332.24	35.75	5.00	1.41	53.00	47.75	1.99
50%	0.48	12340.50	47.00	10.00	1.56	68.00	68.00	3.86
75%	0.65	12396.32	52.25	21.00	1.64	87.25	78.00	7.15
max	6.27	12423.08	59.00	52.00	1.80	98.00	99.00	29.50

Figure 6: Descriptive statistics on 100 best solutions, after 37000 runs with random parameters.

Questions

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

Absolutely! It usually finds the optimal tour. See figure 11.

> For both 10 and 24 cities: How did the running time of your GA compare to that of the exhaustive search?

It's much faster. Se figure 11.

> How many tours were inspected by your GA as compared to by the exhaustive search?

I've not implemented an actual counter, but since the number of possible cycles of length 24 is on par with the number of atoms on earth, it's considerably fewer.

Hybrid Algorithm (IN4050 only)

I did implement Lamarckian selection, as an exercise for myself. Since I'm not a master student I'm not going to elaborate, but I did get slightly better results, at the expense of computation time.

Ending note

I'm fairly certain the best 24 city plan is the one in figure 12.

Bibliography

Eiben, A. E., and J. E. Smith. 2015. *Introduction to Evolutionary Computing*. 2nd edition. Natural Computing Series. Berlin, Heidelberg: Springer, Berlin, Heidelberg.

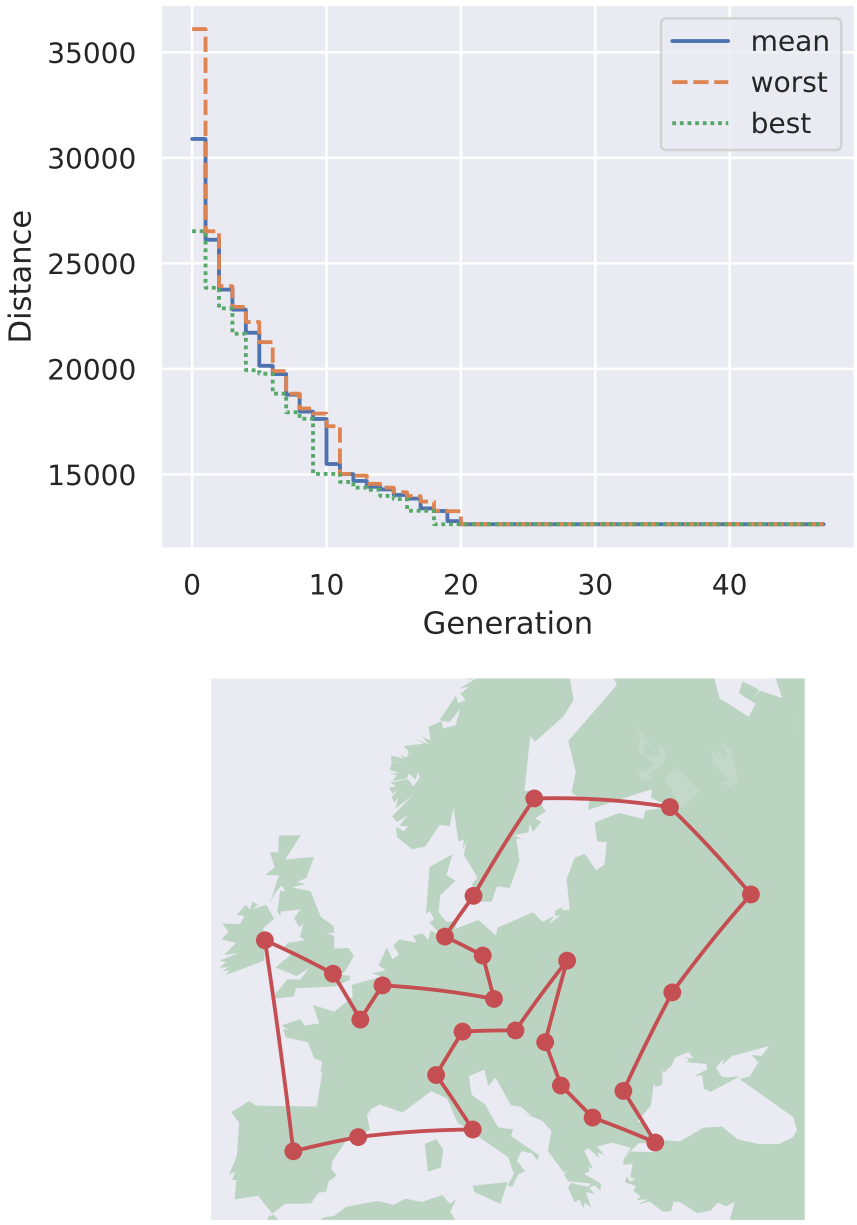

Figure 7: Generational plot of a sample run of the genetic algorithm. Population size 50.

	Size 10		Size 50		Size 100	
	Time	Distance	Time	Distance	Time	Distance
mean	0.52	12892.66	0.60	12748.01	0.63	12632.20
std	0.07	329.33	0.08	377.86	0.10	248.26
min	0.42	12334.35	0.50	12287.07	0.52	12287.07
max	0.63	13675.68	0.81	13329.27	0.87	13277.20

Figure 8: Descriptive statistics on a sample of 20 runs of the genetic algorithm, for different population sizes. 24 cities.

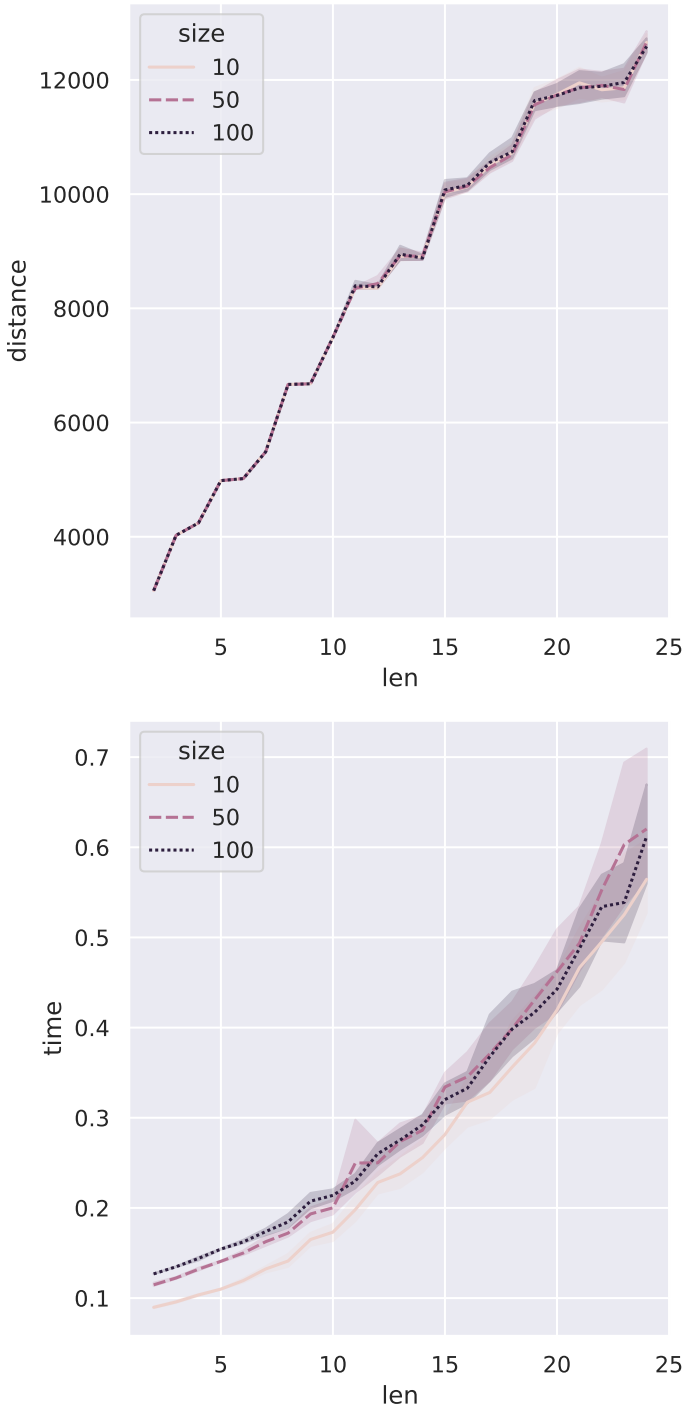

Figure 9: Plots of time and distances for population sizes 10, 50, and 100.

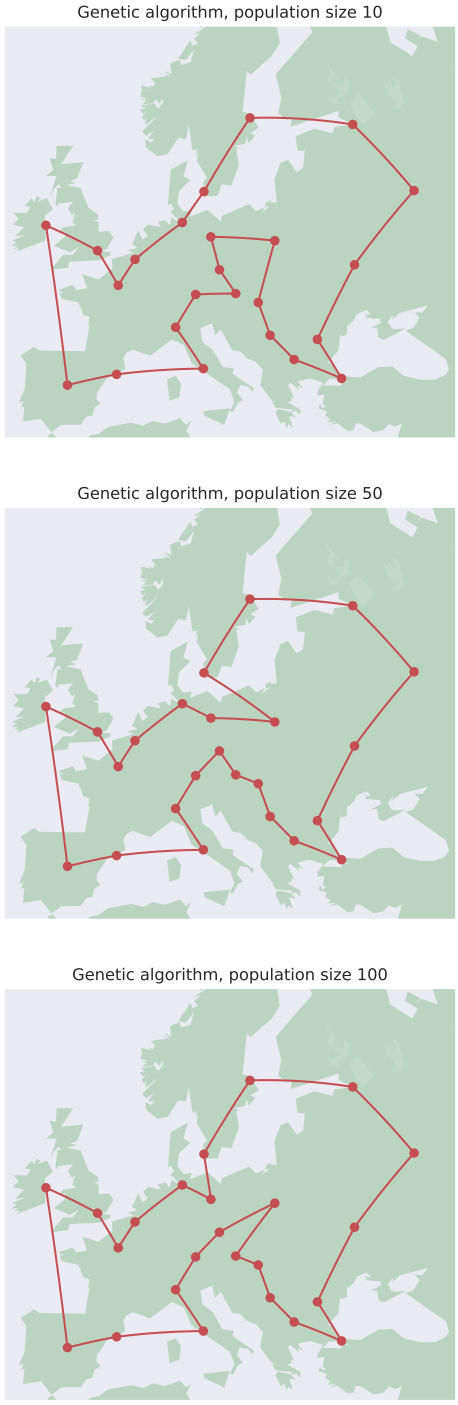

Figure 10: Best plans found in 20 sample runs for different population sizes.

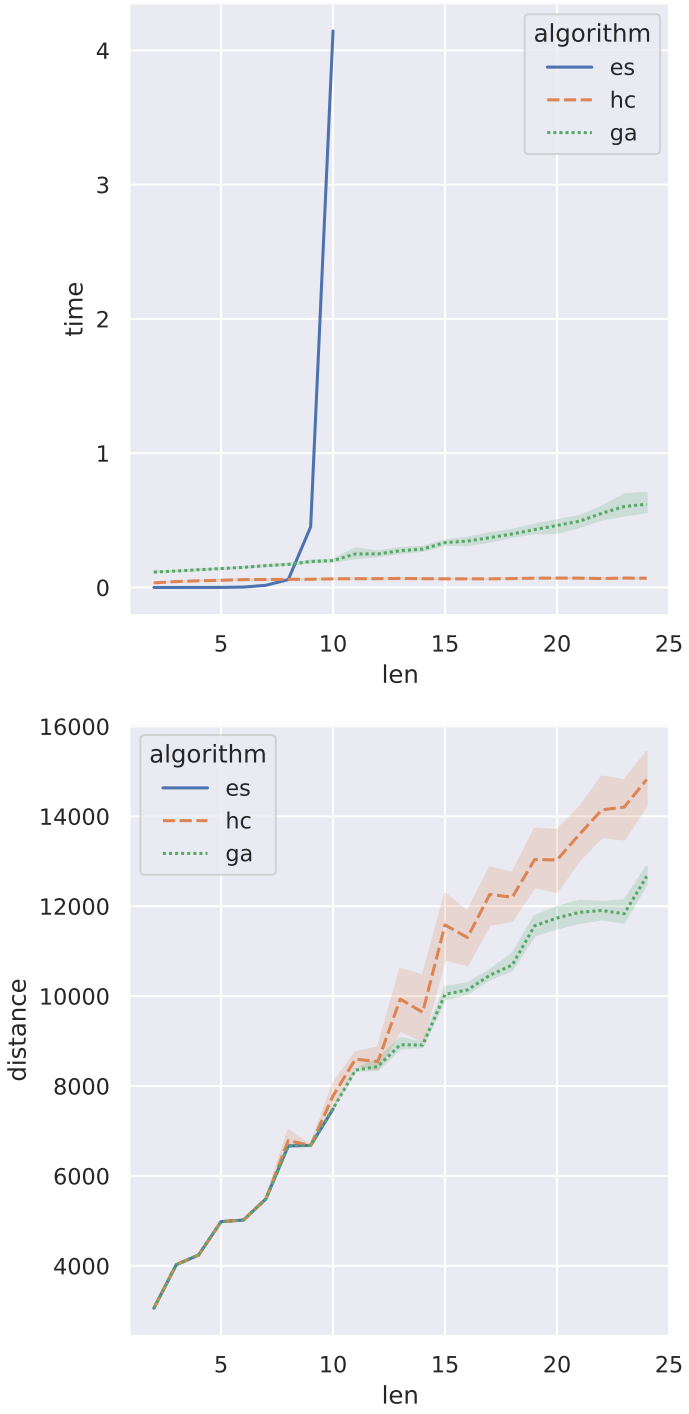

Figure 11: Algorithms compared.

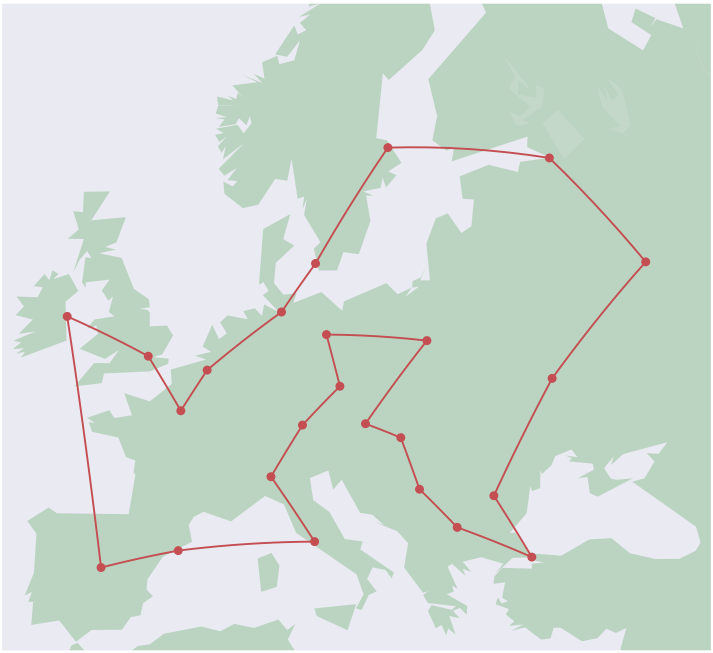

Figure 12: Best plan. Probably.