# Exercise 4: Simulated Annealing

Due date: *November 6th, 2025 (push to GitHub repo before the lecture)*
**General Requests:** *Same as before.*

## 4.1 Simulated Annealing

We talked a lot about continuous domain problems recently. This time, we'll look at one of the most classical combinatorial (discrete) optimization problems: the Travelling Salesman Problem (TSP). As the algorithm was derived (in part) for exactly this type of problem, we'll use Simulated Annealing now [1, Chapters 9 and 18] .

- use this example dataset. it contains generic information on 30 Austrian cities (find their details and coordinates in city_coordinates.json), as well as route information (the specific route one would take by car to go from one city to any other)[1].
- as a short reminder from graph theory: in the TSP, we want to visit all cities exactly once, then return to the city of origin.
- importantly: start with only a few of the cities available until you know your implementation works; otherwise you'll burn a lot of time on nothing.
- implement a Simulated Annealing algorithm as discussed in the lecture. think about how to represent your solution candidates (perhaps you enumerate the cities in a permutation encoding, perhaps something else). the "energy" function in Simulated Annealing just corresponds to the objective function in any other algorithm; think about what objective function to use for this type of problem[2]. use a cooling strategy you find promising.
- use a neighbor generation (mutation) strategy that makes sense for your problem, e.g., any of the mutation strategies we discussed for evolutionary algorithm variations. inversion mutation usually works quite well for TSP problems, but feel free to try out other options as well.
- visualize and update your route solution after each iteration to track how your solution candidate improves. for this, you can use this visualization function. if you run it, you'll see a demo visualization of a short route (Graz-Vienna-Linz- Salzburg-Graz) on an Austria map, to keep things entertaining. if you find nicer ways to visualize everything, feel free to implement your own solutions!

## 4.2 Analysis and Reporting

Add a streamlit page on Simulated Annealing to the documentation you created for the last exercises.
Your final documentation webpage should contain the following sections:

1. Introduction - Overview of the algorithm: basics, strengths/weaknesses, complexity, differences to other algorithms we looked at so far...
2. Methods - Describe your implementation: pros/cons, relevant parameters/functions, critical design choices, ...
3. Results - Display your route solution after every iteration to see how the algorithm converges towards the best route (see last point above), optionally (if you implemented more than one option), allow for different settings of initial temperature or cooling strategy.
4. Discussion - Analyze your findings: expected versus unexpected results, solution quality, efficiency, limitations, possible improvements, ...

The documentation doesn't need to be absurdly long, but it should give a comprehensive overview of the topic that you can use later on to study for the exam.

## References

[1] D. Simon, Evolutionary Optimization Algorithms. Wiley, 2013. [Online]. Available: https://research-1ebsco-1com-1195qzf320241.perm.fh-joanneum.at/c/kofjhs/search/details/4sh2uyq6wn?db=nlebk&db=nlabk

[1] Note: the routes do not distinguish 'to' and 'from'; i.e., the route from Villach to Feldkirch is not stored again if the route from Feldkirch to Villach has already been stored previously.
[2] What do you want to maximize or minimize in your problem?