Optimization laboratory: Traveling Salesman Problem

Léa Ricard

Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
École Polytechnique Fédérale de Lausanne

9 April 2025

EPFL

Laboratory schedule

Date	Topic
9.04.2025	Optimization laboratory
16.04.2025	Multi-objective laboratory
	Optimization project
30.04.2025	Optimization project
07.05.2025	Optimization project
21.05.2025	Optimization project

Traveling Salesman Problem (TSP)

Problem definition:

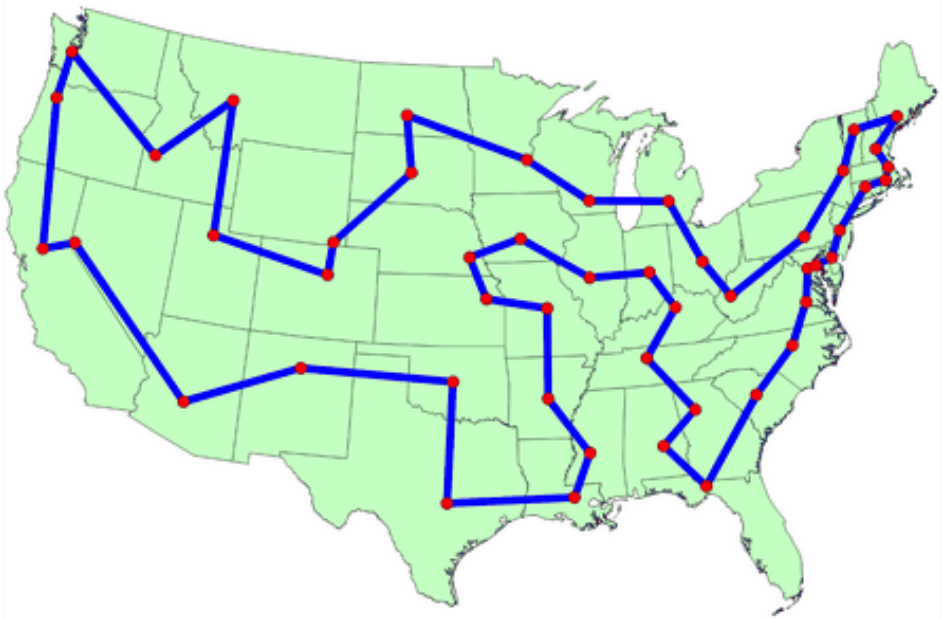
- A salesman must visit n cities.
- Every city must be visited exactly once.
- The salesman starts and ends the trip at their home city.
- The total trip length is assumed to be the cost of the travel.

Objective

- What sequence of cities minimizes the travel cost?

Traveling Salesman Problem (TSP)

- Applications:
  - Vehicle routing;
  - Job shop scheduling;
  - Computer wiring;
  - Etc.
- Largest instance solved by Concorde's TSP solver: 85,900 cities.



Traveling Salesman Problem (TSP)

Problem encoding

We consecutively number the cities: $0, 1, ..., n$.
We encode the solutions as $x = (x_0, x_1, ..., x_n, x_0)$ where

- x_0 is the index of the home city,
- x_i is the index of the i^{th} city visited along the way, and
- x_n is the index of the last city visited before returning home.

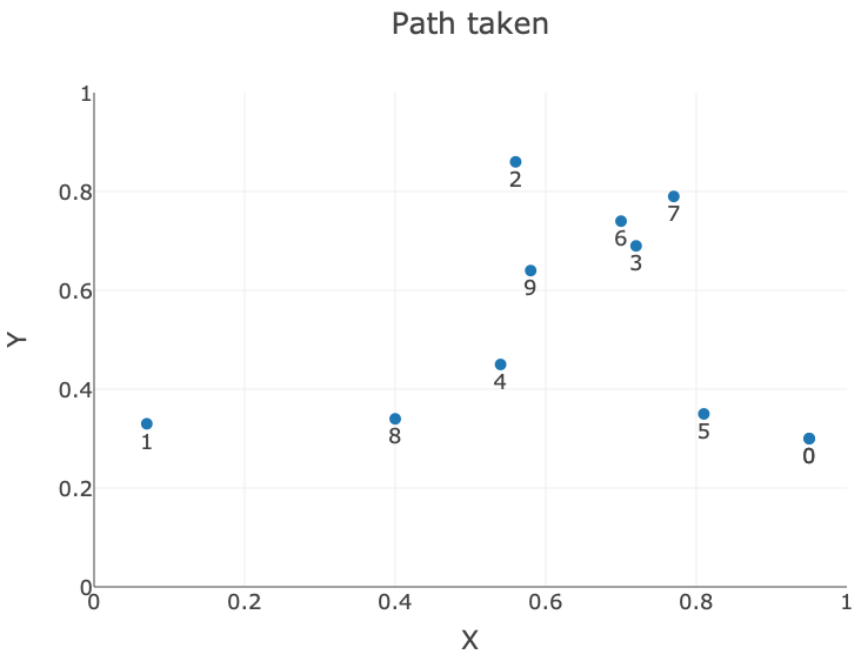
Implementation functions

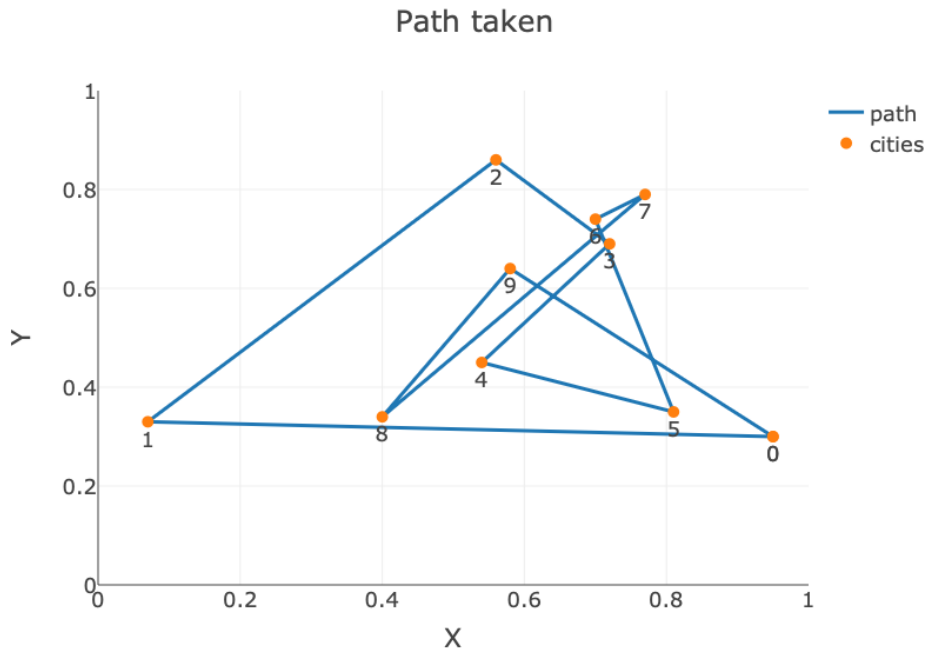
Core functionality

`simulate_cities(seed, n_cities)`

`draw_salesman(path, cities)`

`evaluate_city_sequence(path, cities)` → Calculate the total distance traveled





Algorithms to implement

Full enumeration:

- Understand its limitations

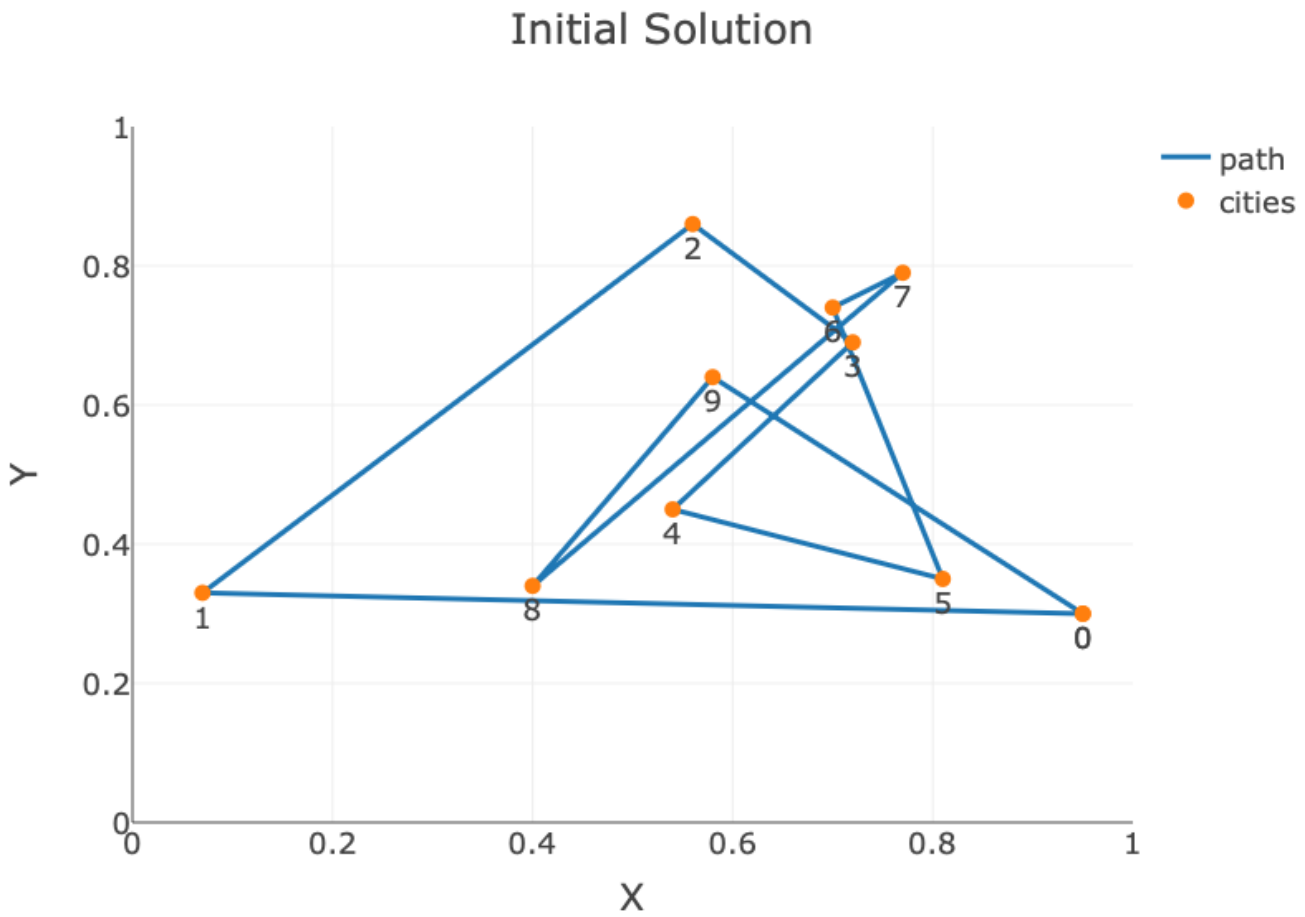
Optimization algorithms:

- Greedy algorithm
- Local search
- Variable neighborhood search
- Simulated annealing

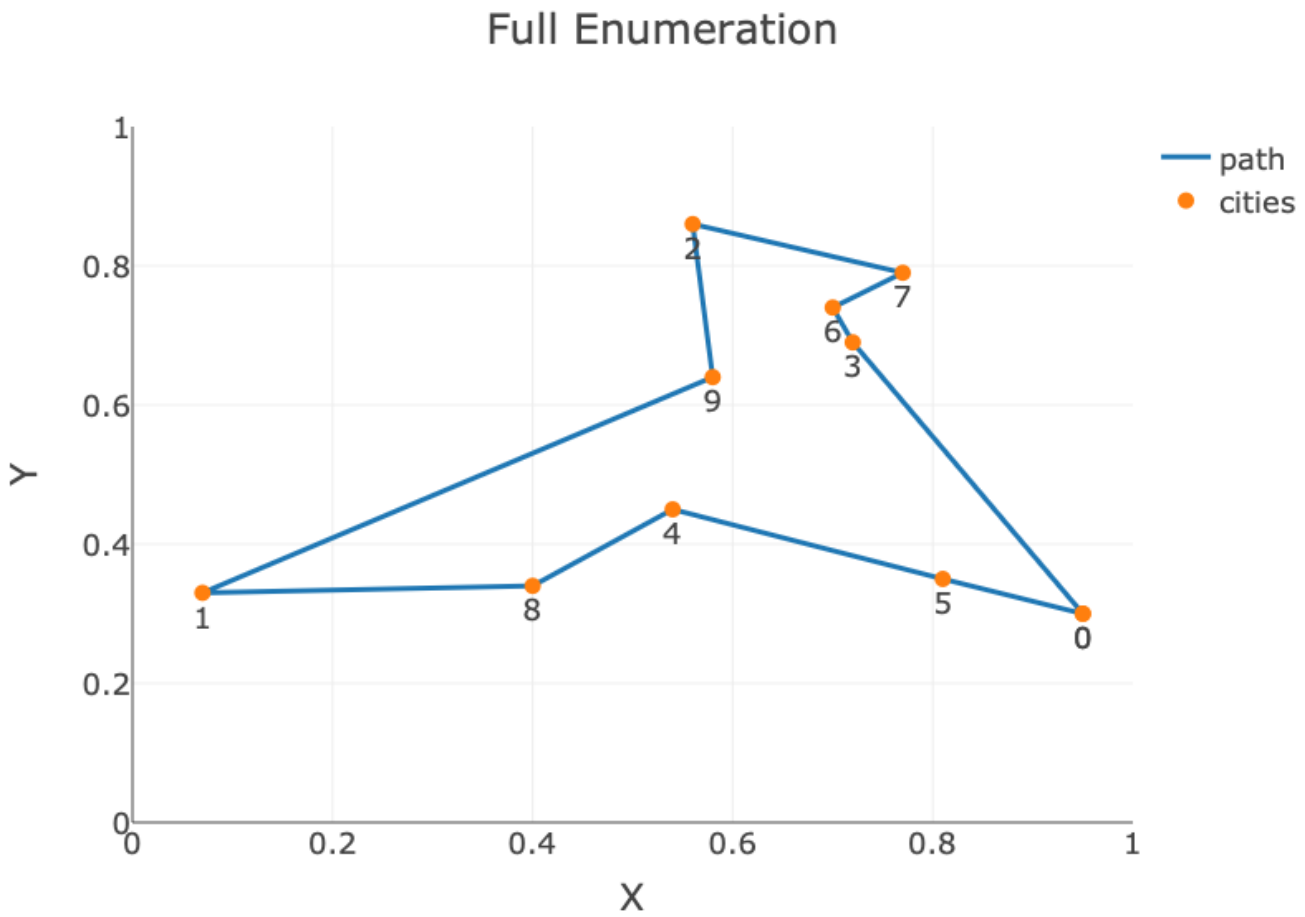
IMPORTANT

- The suggested framework is only a **suggestion**!
- Feel free to organize the code in the manner you find the most appropriate!

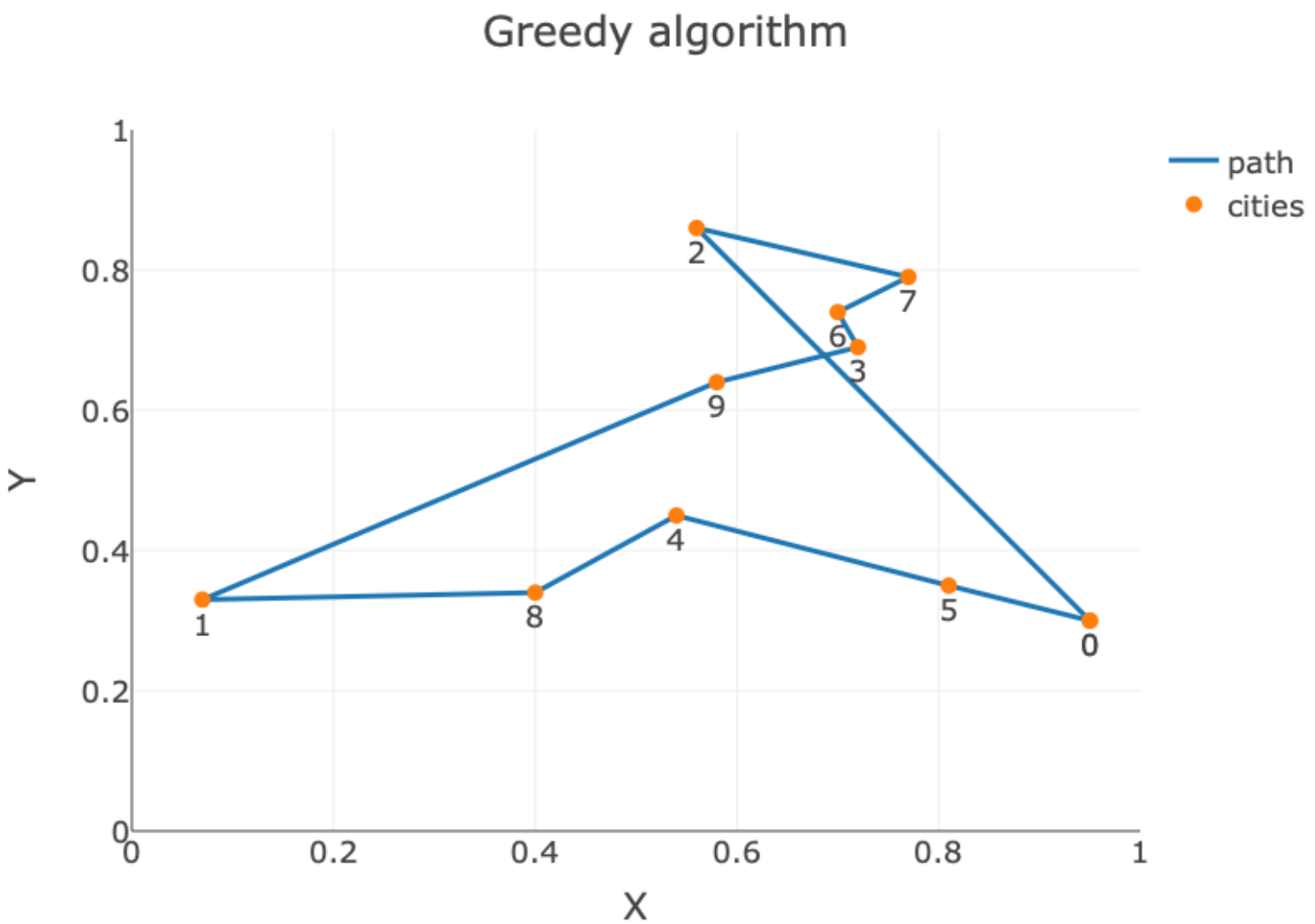
My results - An initial solution



My results - Full enumeration



My results - Greedy algorithm



My results - Local search (LS), variable neighborhood search (VNS), and simulated annealing (SA)

- Same solutions as full enumeration (optimal).
- Computing time for instances with 10 cities and using the parameters specified for the SA algorithm: $\text{CPU}_{\text{LS}} < \text{CPU}_{\text{VNS}} < \text{CPU}_{SA}$.

My results - Simulated annealing