Applied Computational Intelligence Project Report

David Gao Xia nº 99907 , João Barreiros C. Rodrigues nº 99668

Oct. 2024

Introduction

This project focuses on solving the Traveling Salesman Problem (TSP) using Evolutionary Computation methods. The goal is to optimize travel across 50 major European cities via collective transport (plane, train, bus), minimizing travel costs or travel time while ensuring each city is visited exactly once.

Data Collection

The cities selected for our dataset are detailed in the xy.csv file. We gathered data for the plane datasets from Google Flights. For the bus datasets, we utilized BusBud. Finally the Train dataset information was collected from Omio.

Single-Objective Genetic Algorithm (SOGA)

We tackle the Traveling Salesman Problem (TSP) by using Evolutionary Computation with Elitism, were we keep the best individuais on the population instead of replacing it with just the offsprings.

In cases where we need to find the minimal cost of time for a single type of transport, our population individuals are represented as lists of integers, with each number corresponding to a city in the xy.csv file. The search space has a size of #NumberOfCities!. We employed ordered crossover and shuffle indexes for mutation to ensure that each city is visited exactly once. For selection, we utilized a tournament method of size 3.

In cases where all types of transport (plane, bus, train) are utilized, our population individuals are represented by two lists: one list of integers, similar to the case with a single transport type, and another list of strings ["plane", "bus", "train"], indicating the mode of transport between two cities. The search space has a size of (#NumberOfCities! * 3^#NumberOfCities). We employed ordered crossover for the list of integers and uniform crossover for the transport type list. For

mutation, we used shuffle indexing for the integer list and selected a random transport type for the other list. For selection, we utilized a tournament method of size 3.

Results

#Cities	Plane mCost Mean	Plane mCost STD	Bus mCost Mean	Bus mCost STD	Train mCost Mean	Train mCost STD
30*	2040.03	236.85	894.3	392275.16	1864.2	629204.29

#Cities	Plane mTime Mean	Plane mTime STD	Bus mTime Mean	Bus mTime STD	Train mTime Mean	Train mTime STD
30*	2197.83	163.12	16814.06	379934.64	7997.4	627572.86

#Cities	EveryTransport mCost Mean	Every Transport mCost STD	Every Transport mTime Mean	Every Transport mTime STD
10	336.0	7.34	710.0	8.82
30	833.9	180.91	2005.77	223.70
50	2202.17	866871.54	3868.27	820750.11

It is important to note that due to our train dataset being fairly sparse for the Train only tests, the number of cities targeted for it where 23 instead of 30, to avoid choosing invalid solutions.

Looking at the results, it's clear that the cost-effective choice is prioritizing bus transport, while in the time minimization problem the plane is the preferred method of transportation. The results emphasize the trade-off between budget and speed. The dataset's sparsity led to numerous invalid solutions during algorithm execution, contributing to the high standard deviation.

Additional Heuristic

Our heuristic picks cities on a map by starting with the highest Y-coordinate on the left half, moving downward.

Once finished, it switches to the right half, starting with the lowest Y-coordinate and moving upward. To maintain diversity, we divide the population by HEURISTIC_SIZE % of the total population and randomly select the rest.
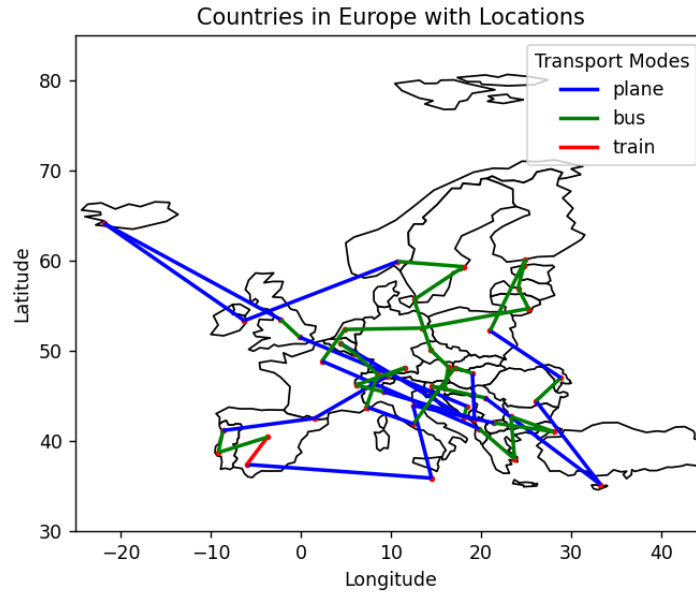
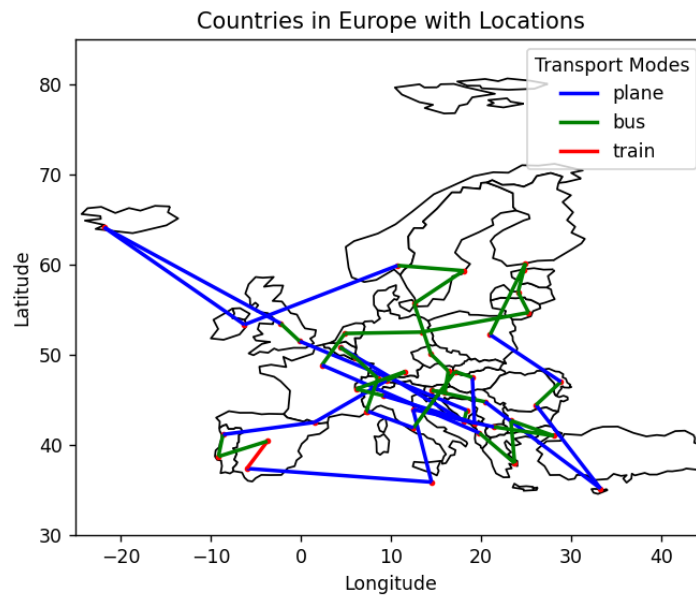Figure 1: Fitness evolution for Cost Minimization without Heuristics, cities=30

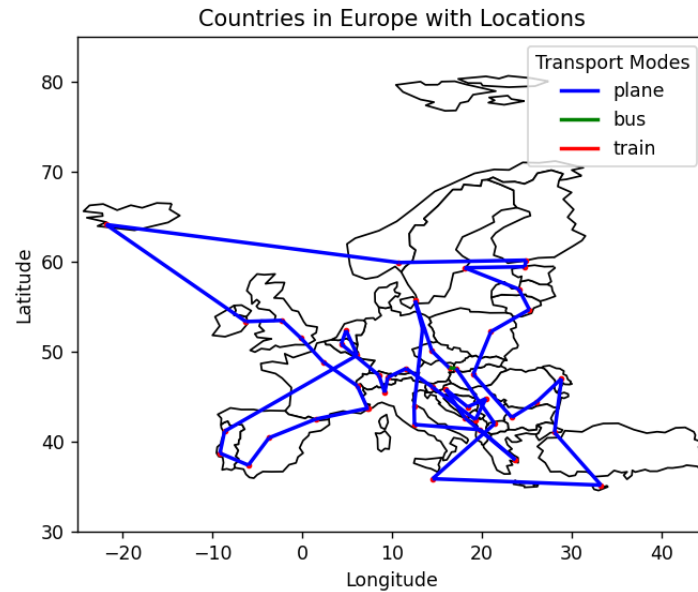Figure 2: Chosen Tour for All-Transport, Cost Minimization without Heuristics, cities=50

Figure 3: Chosen Tour for All-Transport, Time Minimization without Heuristics, cities=50

Despite applying heuristics, the results didn't show significant differences. This likely stems from our dataset values not correlating directly with city distances, making our heuristic solution less effective for this problem.

#Cities	EveryTransport mCost Mean	Every Trasnport mCost STD	Every Trasport mTime Mean	Every Trasport mTime STD
10	336.0	7.87	710.0	9.37
30	876.73	182.83	2046.4	251.84
50	2158.4	866437.79	3778.12	820512.66
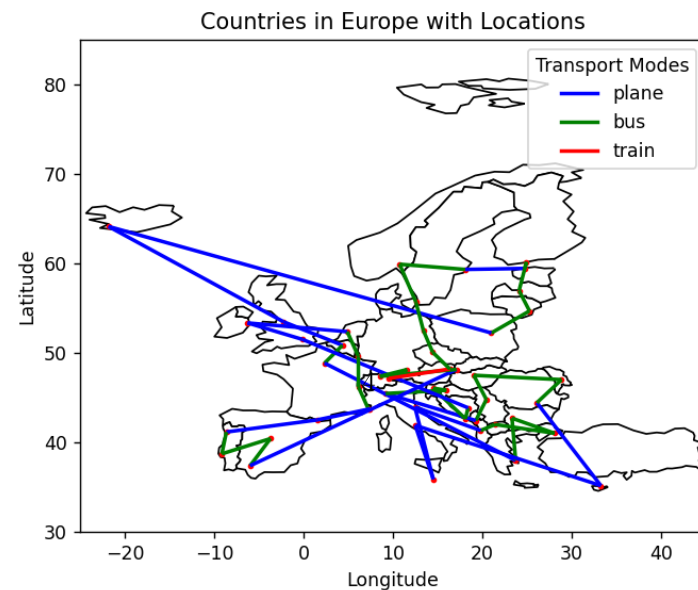
Multi-Objective Genetic Algorithm (MOGA)

Our MOGA, has the same foundation of our SOGA, using a eaSimple variation with Elitism, with the option to use the same heuristic described before.

To fit the multi-objective problem we had to changed some components of our algorithm most notably: - Fitness registry and calculation/evaluation - Selection - Solution extraction from result population

The Fitness functions now had to support two objectives, both to be minimized. To achieve this we simply return a tuple from the evaluation function composed of both the respective monetary cost and time taken for a solution.

The Selection is now made using the NSGA-II algorithm:

Finally for the solution extraction from the result population we first calculate two points for the final generation: the ideal point (the minima observed in the possble solutions for both objectives) and the centroid (the average of the non-dominated front solutions). By finding the solution that minimizes the vector distance to both points we extract a middle-ground solution that minimizes both monetary cost and time taken.

Results

We could not generate valid solutions for our dataset when using the MOGA, we suspect that this is once again due to the sparsity of our dataset. Therefore the results presented in this section are taken from the dataset given by Professor Horta.

#Cities	Cost for MinCost Solution	Time for MinCost Solution	Cost for MinTime Solution	Time for MinTime Solution
10	910.44	49h	1440.27	12h51m
30	2954.71	185h43m	4740.34	46h
50	3545	170h56m	5007.63	54h16m

Notably our MOGA significanly benefited from Heuristics, for which we chose to maximize the respective parameter to 100%.
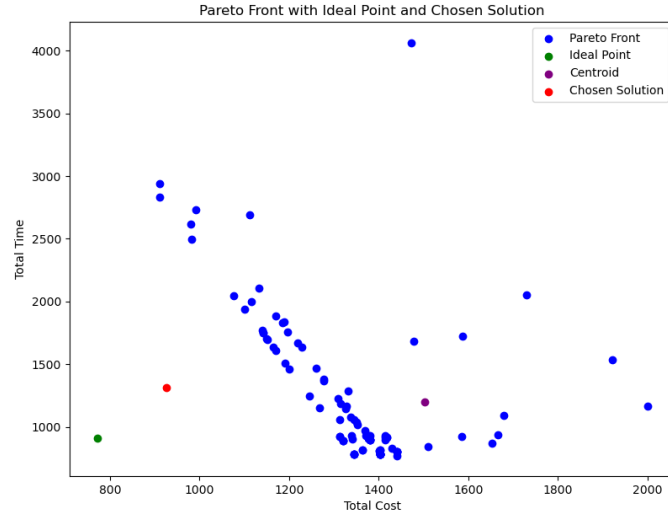
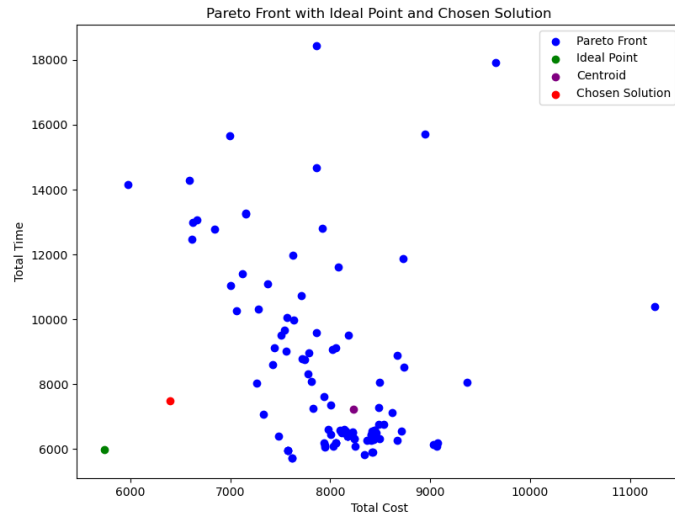Figure 5: Last generation Pareto front for cities=10

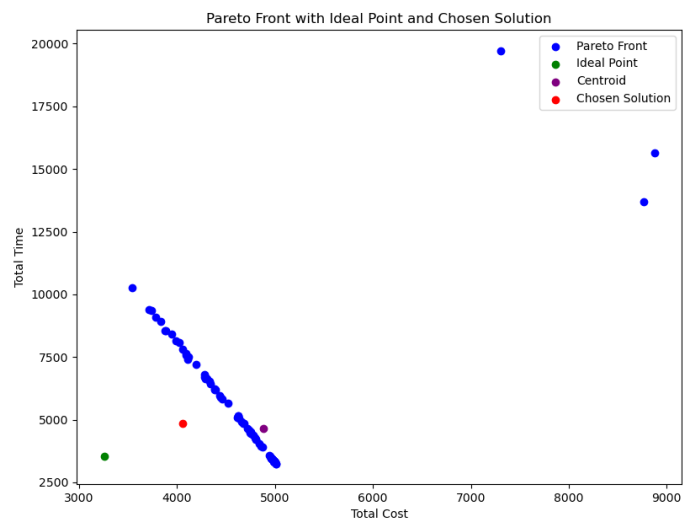Figure 6: Last generation Pareto front for cities=50 and without Heuristics

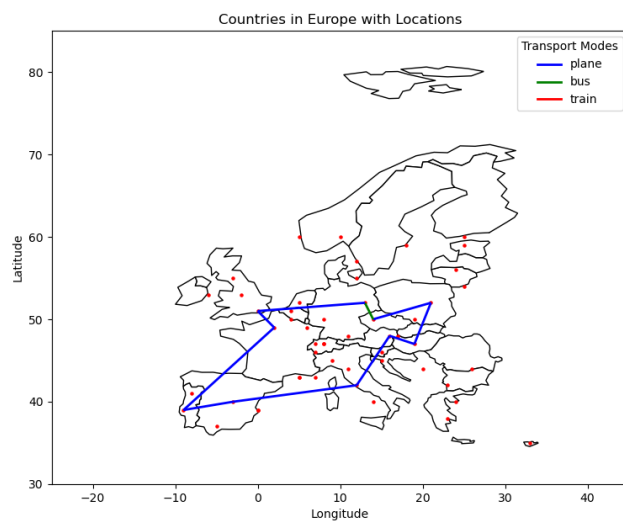Figure 7: Last generation Pareto front for cities=50
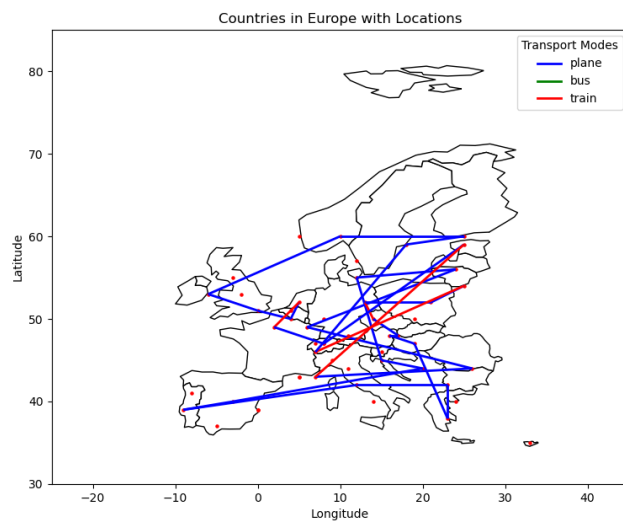

Figure 8: Chosen Tour for cities=10

7

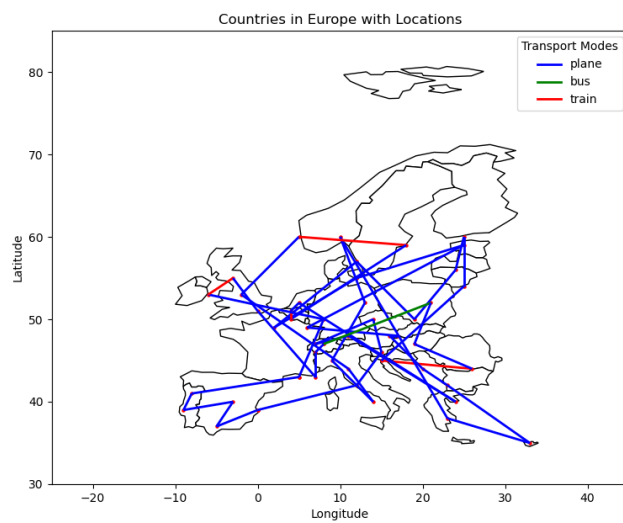Figure 9: Chosen Tour for cities=30

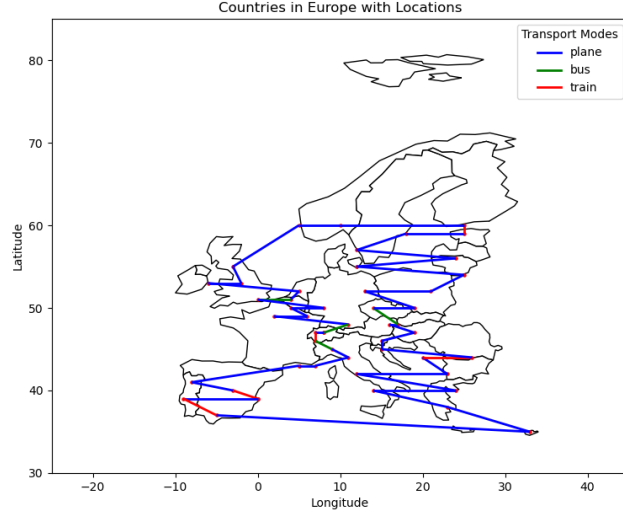Figure 10: Chosen Tour for cities=50, without heuristic

8

Figure 11: Chosen Tour for cities=50

Conclusion

Our main indrence found was the created dataset. Due to its sparsity and due to the method chosen for representing invalid solutions (with soft penalties instead of hard penalties) convergence for valid solutions was not always achievable.

After using the MOGA with Heuristics with understood that the minimal differences that our heuristic solution was producing were indeed related to our dataset, since it performed distinctly well for the other dataset.