ARTIFICIAL INTELLIGENCE

G32 2022/2023

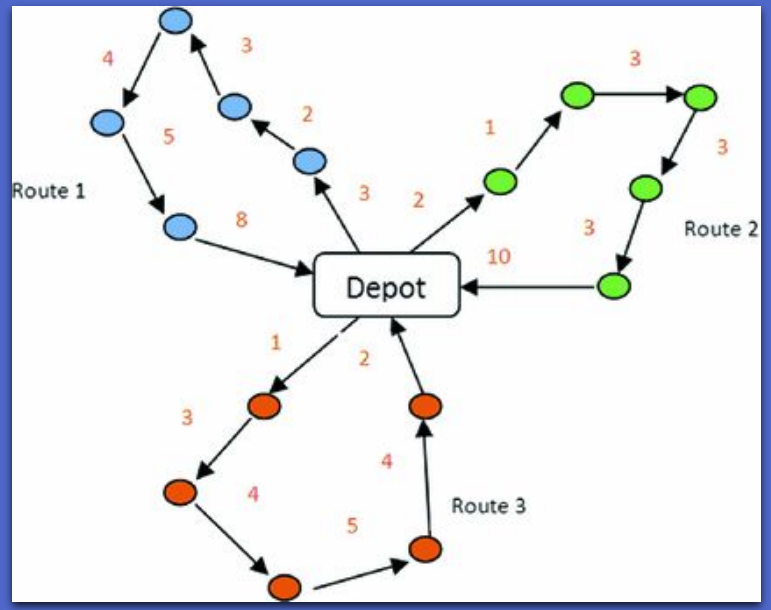
Metaheuristics for Optimization/Decision Problems

Diogo Babo - up202004950@up.pt
Gustavo Costa - up202004187@up.pt
João Oliveira - up202004407@up.pt

RELATED WORK

The ASAE's optimal route inspection problem is similar to the vehicle routing problem, though the constraints may be different.

This problem was presented in a thesis here at FEUP, the thesis can be found here.



PROBLEM DESCRIPTION

ASAE inspection route problem – 3A

This problem's scenario consists of a city with different establishments that must be inspected by the ASAE vehicles.
All the vehicles depart from the same depot at the same time and the goal is to optimize the vehicles route so that we **minimize** the amount of time taken to inspect all the establishments.

PROBLEM FORMULATION

Goal:

- Minimize time taken to inspect all establishments

Solution Representation:

- List of paths - {v1: [e0, e2, ..., e0], vn: [e0, e3, ..., e0]} where the index represents the vehicle number and e the establishments visited

Neighborhood/Mutation & Crossover Functions:

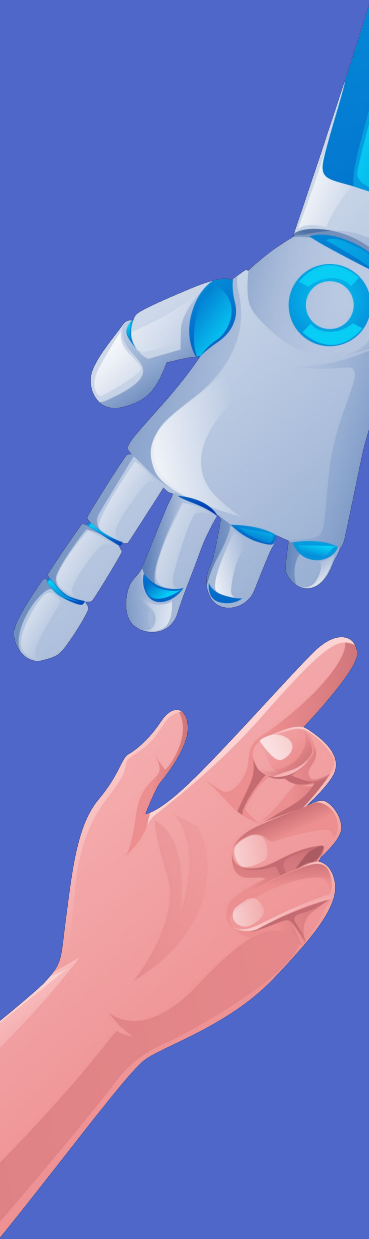
- Generate a initial random and greedy solution (60% inspect. time & 40% distance)
- Switch 2 nodes in the same vehicle, or between two different vehicles
- Remove node from one vehicle and add it on other
- Crossover whose midpoint is given by a heuristic depending on the number of vehicles

Hard Constraints:

- The number of vehicles (n) is 10% of the number of establishments
- Must visit all establishments
- All vehicles depart from the same depot at the same time (09:00)
- All vehicles must return to the initial depot in the end
- Inspections must happen only if the establishment is open else need to wait until it opens

Evaluate Functions:

- Given a solution retrieve the minimum time taken to visit all nodes



WORK DONE

Programming Language:

- Python

Work Developed:

- Parsing of establishment and distance files
- Evaluation Function
- Hill Climbing, SA, Tabu Search & Genetic Algorithm
- Initial Random/Greedy Solution generator
- Solution evolution plot
- Map with the vehicles' path

Development Environment:

- PyCharm / VSCode

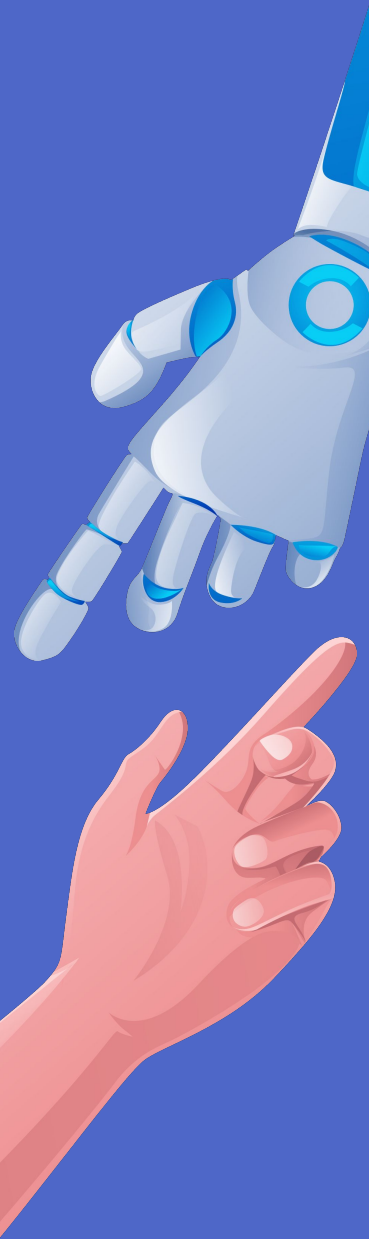
Data Structures:

- Class for representing an Establishment
- Graph to represent the city, consists on a list of nodes (establishments) and an adjacency matrix where edges[src][dest] = Time from source establishment to the destination one
- Dictionary for Tabu Memory

APPROACH:

- Our evaluation function, **Evaluate_solution(city: City, solution, eval_func)**, returns the total time taken to inspect all the establishments, which is what we are trying to minimize. Basically it loops through all the vehicles and returns the time taken by the very **last one** to arrive at the depot.

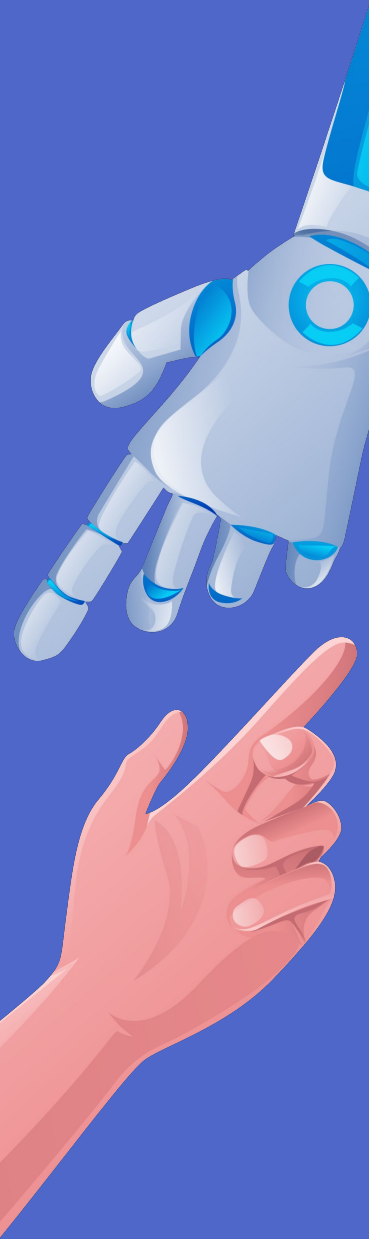
- We have six different neighbor/mutation functions:
  - Remove a random establishment from the worst/random vehicle and add it into a random position at other random vehicle.
  - Switches two 2 random establishments between a worst/random vehicle and a random one.
  - For the worse/random vehicle, changes the order of 2 establishments in the path.



HillClimbing & SA:

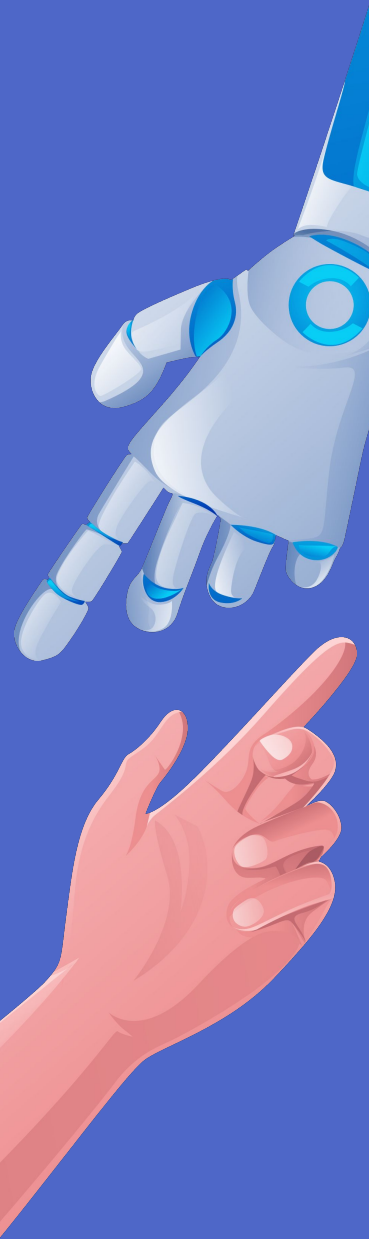
- Hill Climbing:
  - In each iteration apply a mutation to the current best solution.
  - Proved to be a fast and efficient algorithm.

- Simulated Annealing:
  - Similar to the hill climbing algorithm, but we accept worse solutions based on the temperature in order to escape local minimums. Meaning, the smaller the temperature the lower it is the probability of accepting worse solutions.
  - In each iteration we use a cooling schedule in order to decrease the temperature (Decreases by a random amount between 0.001% and 0.005%).



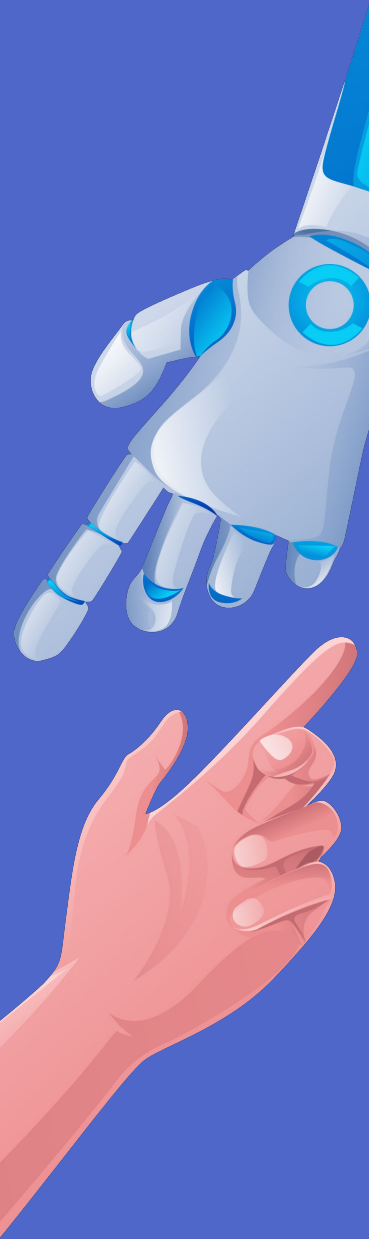
Tabu Search:

- In our implementation of TS we always allow “Best Improvement”, but in order to escape local minima we ignore solution that may have been visited before.
- Since storing the entire solution at each iteration would very inefficient, we decided that an appropriate **Tabu Criteria** would be to store the **ID** of the establishments that were changed within the mutation.
- To achieve this we have a dictionary where the **keys** are the IDs and the **values** are number of iterations that must occur so that the establishment can be used again. (cooldown)
- This cooldown (**Tabu Tenure**) is currently a fixed number, since it is what worked out for us the best.



Genetic Algorithm:

- Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation.
- We generate two offspring through a classical crossover, by swapping a portion of both parents (the size of the portion is randomly chosen depending on the number of vehicles).
- We start by generating an initial solution, either randomly/greedy, and in each iteration we:
  - Generate two offspring, being one of the parents the fittest and the other chosen through roulette selection;
  - Depending on a probability (50%) we mutate those two offspring;
  - Depending on the **reproduction_rate**, we also randomly generate new offspring in order to add diversity to the population itself.
  - In the end, we remove the less fit species of the population, making it start with a fitter population the next time.



Results:

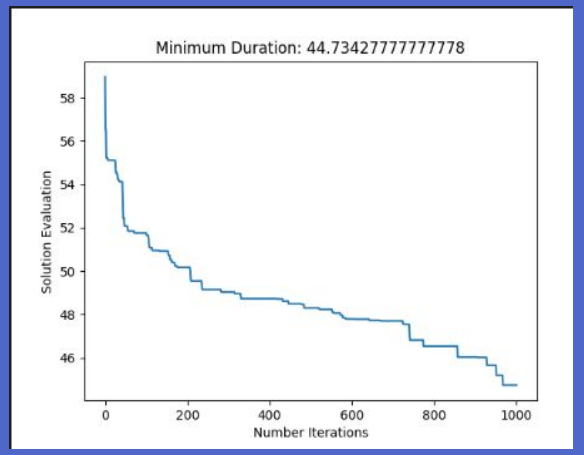


Fig 1 - Hill Climbing

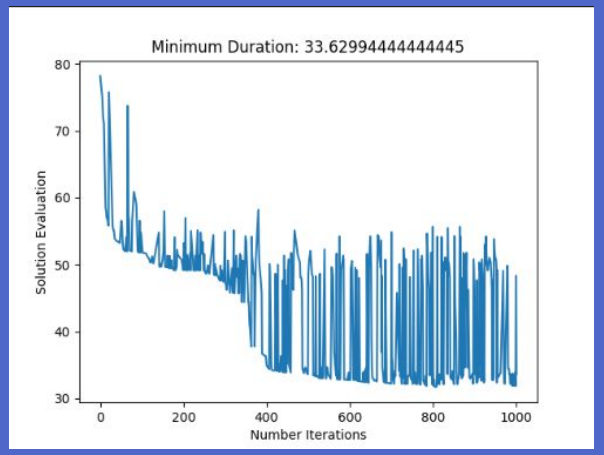


Fig 2 - Simulated Annealing

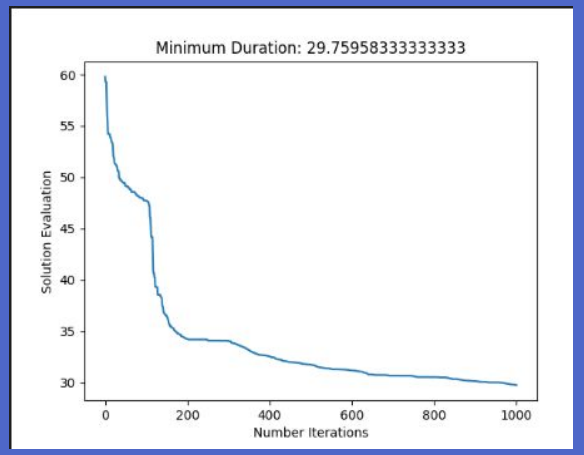


Fig 3 - Tabu Search

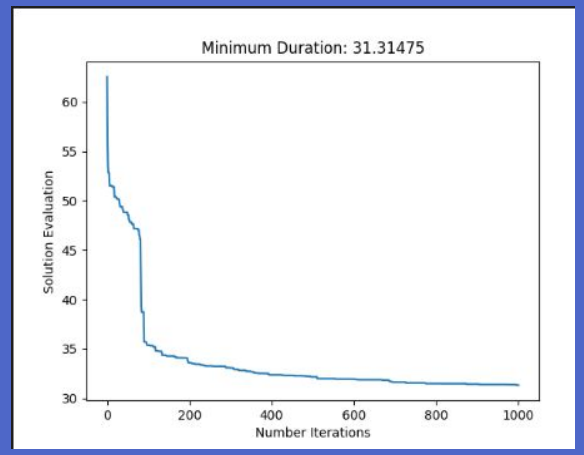


Fig 3 - Genetic

All the results were taken with the following metrics:

- City Size - 1000
- Number of Iterations - 1000
- Initial Solution - Greedy (Better to compare algos.)
- Neighbor Function - **neighbors_2**
- SA Initial/Min Temperature - 150/1
- Neighborhood Space Size - 20 (TS & Genetic)

Alg	Score	AVG	Improvement
Hill	Start	50,48611111	40,94985786
	Final	29,81212037	
SA	Start	50,48611111	35,66994956
	Final	32,47774074	
Tabu	Start	50,48611111	50,59319578
	Final	24,94357407	
Genetic	Start	50,48611111	39,25228794
	Final	30,66915741	

Fig 1 - Solution Improvement

CONCLUSION:

- Several approaches were implemented and tested as part of our work. The desire for a better result drove us to experiment with various heuristics, algorithms, and parameters, as well as different neighbourhood definitions.

- The results went according to what we thought in theory apart from the **Genetic** algorithm, which we thought would find the best optimal solution among all the algorithms. However, this didn't happen as **Tabu Search** managed to find a better solution with a reasonable execution time.

- We consider this project to have been a positive introduction to artificial intelligence, prompting us to dive deeper into the topics addressed.