U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Optimal Inspection Routes Problem

Inspected establishments maximization

João Pinheiro (202008133)
José Araújo (202007921)
Ricardo Cavalheiro (202005103)

01 Problem Specification

02 Work Implemented

03 Problem Formulation

04 Implemented Algorithms

05 Experimental Results

06 References

Problem Specification

- This problem is a simplified version of the actual ASAE (Portuguese Economic and Food Safety Authority) inspection route problem and is an instance of the Vehicle Routing Problem;
- Given a finite number of vehicles, the maximum number of establishments must be inspected within an operating time interval (8 hours), finding an optimal solution;
- Inspections must align with the establishment's opening hours and can only begin if it is open
- If an inspection has started, it can be completed even if the duration of it surpasses the operator's closing time
- Number of vehicles available is proportional to the number of existing establishments

No. Vehicles ≈ No. Establishments * 0.1

Work Implemented

- **Programming language**

  C++

- **Dev. Environment**

  VSCode and CLion

- **Data Structures**

  C++ classes, to represent graph and other objects (OOP)

- **Data representation**

  Graph reused from previous curricular unit (AED - Algoritmos e Estrutura de Dados, @FEUP) and accordingly adapted to suit our problem

  MatPlotLib C++ library to represent map and plots

- **Implementation of optimization algorithms:**

  Hill Climbing Simulated Annealing
  Tabu Search Genetic Algorithm Iterated Local Search

Problem Formulation

- **Solution Representation**

[Vehicle(n): [establishment(m), ..., establishment(p)]], [m, p] $\in$ [0, noEstablishments],

n $\in$ [0, 0.1*noEstablishments]

- **Mutation functions**

Mutation 1: Exchange two establishments in a solution

Mutation 2: Exchange one establishments for one that did not appear in the solution

Mutation 3: Generate a new path between two nodes

Mutation 4: Scramble the order of a subset of nodes in a vehicle’s path

Mutation 5: Remove a node from a vehicle’s path and insert two new nodes

Among all the mutations developed, the fifth is the one that turns out being more efficient and provides better results.

Problem Formulation

- **Crossovers**

  Crossover 1: Select a midpoint, smaller than the shortest path in the solution, which will be dividing the both solutions in the midpoint. The two new resultant solutions consist in changing the cuts of the parents solutions.

  Crossover 2: Take the middle part of the first parent's solution between two crossover points and filling the remaining parts with the nodes from the second parent's solution, creating the child solutions.

- **Evaluation Functions**
  1. Number of visited establishments
  2. Parish transfers efficiency (penalize parish transfers) (XXXYY-> XXX number of visited establishments, YY % of parish transfers)

- **Heuristics**
  - Closest node A from a given node B, having A ≠ B, A not visited

- **Hard Constraints**
  1. Each vehicle's route must not exceed 8 working hours
  2. Number of vehicles must correspond to approximately 10% of the number of existing establishments
  3. All vehicles leave at the same time and place, they also must finish their routes in the starting point.

Implemented Algorithms

- **General parameters applied to algorithms**
  Iteration number: 1000
  Mutation function: 6 (random mutation then 5 is applied on top)
  Evaluation function: 2 (parish transfers efficiency)

- **Hill Climbing**
  Simple optimization algorithm that starts with an initial solution and iteratively improves it by taking steps in the direction of the steepest uphill slope. As expected, as it cannot escape local optima, solutions are the worst among all optimization algorithms implemented.

- **Iterated Local Search**
  We implemented a very simple version of this algorithm, running Hill Climbing a number of times where each new iteration initial solution is the final solution of the previous iteration.

  Parameters:
  Number of iterations: 10

- **Simulated Annealing**
  Variation of the Hill Climbing algorithm and is designed to overcome its limitations, which can get stuck in local optima. As expected, could escape local optima problem from previous algorithm, so the results were better, though not better than Tabu Search algorithm.

  Parameters:
  Cooling schedule: 0.999
  Initial temperature: 1000

Implemented Algorithms

- **Tabu Search**
  Designed to overcome the limitations of both Hill Climbing and Simulated Annealing. Uses a memory structure to keep track of previously visited solutions, and avoids revisiting them in the search process. Overall the algorithm that produced the best results among the ones implemented.

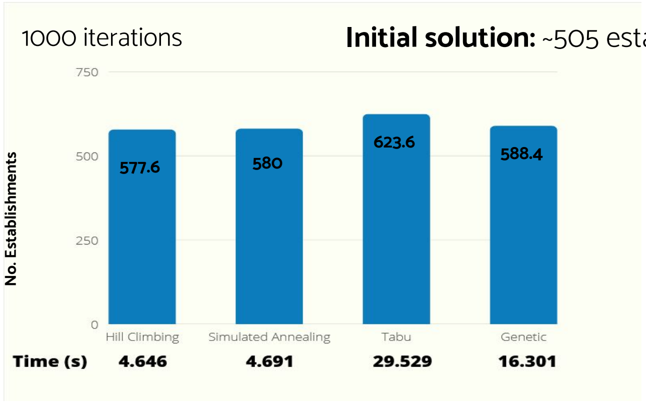
  Parameters:
  Tabu list size: 20
  Neighbourhood size: 4

- **Genetic Algorithm**
  Inspired by the process of natural selection and genetics. Designed to efficiently search large solution spaces and find optimal or near-optimal solutions. Produces overall one of the best solutions, though it does not surpass Tabu's results.

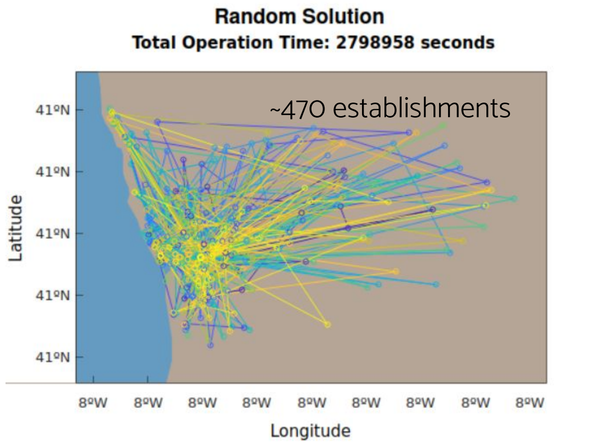
  Parameters:
  Population size: 20
  Tournament size: 8
  Crossover function: 2
  Mutation rate: 90%

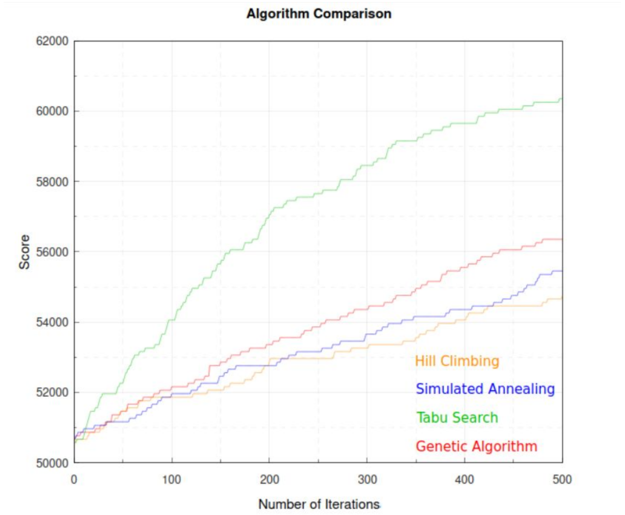
Experimental Results

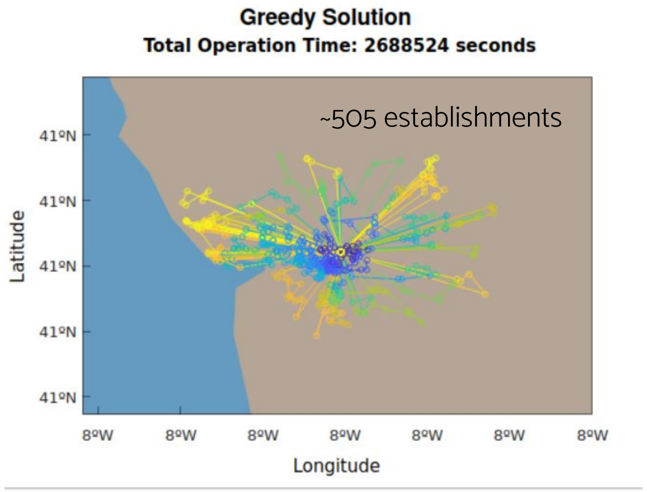
1000 iterations **Initial solution:** ~505 establishments



Time (s)	4.646	4.691	29.529	16.301







References

❑ Previous work done in AED Curricular Unit (Algoritmos e Estrutura de Dados, @FEUP)

❑ A Master Thesis by a student from Universidade do Minho about vehicle routing problems and route optimization (http://repositorium.sdum.uminho.pt/handle/1822/22289)

❑ A Dissertation by a student from Universidade do Porto about a variation of our problem (https://repositorio-aberto.up.pt/handle/10216/137683)

❑ An article on finding the shortest path with learning algorithms (https://www.researchgate.net/publication/267674296_Finding_shortest_path_with_learning_algorithms)