Traffic Light Signaling

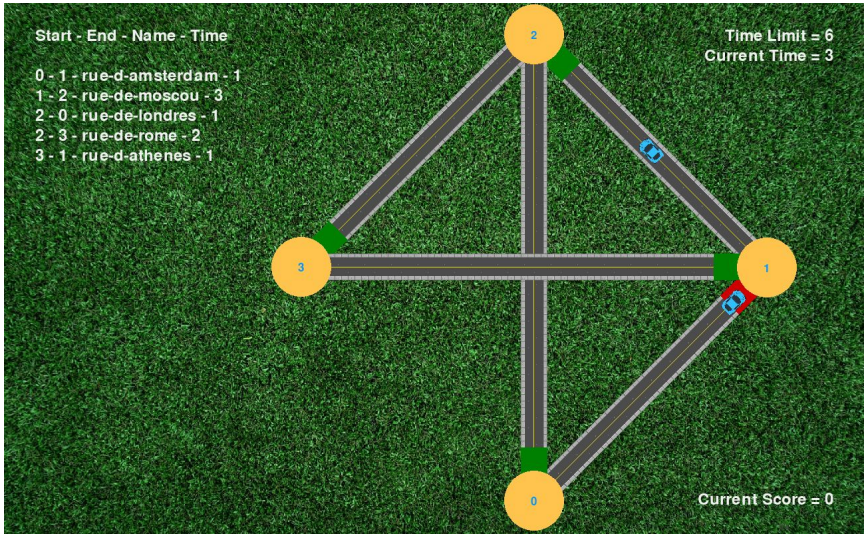
Project 1: Final report
Artificial Intelligence, L.EIC

Group 1, 3LEIC08
Bruno Mendes - up201906166
David Preda - up201904762
Fernando Rego - up201905951

The problem

Maximize the number of cars that reach their destination in time, in a city composed of streets connected through intersections, by changing the schedule where the intersections green lights turn on and off for each incoming street.

The problem's full description can be found here.



Related work

The presented problem served as the qualification round for Google Hash Code 2021. Participants had less than 5 hours to come up with solutions.

We validated our simulation function by comparing the results to a third party's and had a look at community solution proposals:

- A local search method based on Hill Climbing
- Try to make most use of green lights for each iteration; brute force search neighbours

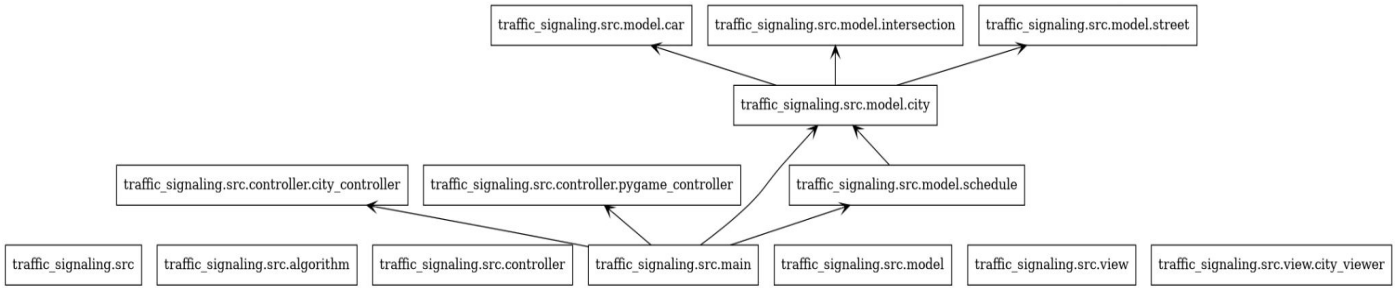
Problem Formulation

- Solution Representation
  - Schedule - {i0: [s1,s2,...,sn], ... ,in: [s1,...,sn]} where i represents each intersection and s each incoming street with the green traffic light on at the nth second
- Neighborhood/mutation and crossover functions
  - Generate a new random schedule for a random intersection
  - Mutate a single street for a single intersection
  - Join intersection cycles from two different schedules
- Hard constraints
  - Can only schedule green lights for intersection incoming streets
- Evaluation Function
  - Car path simulation
  - Calculate simulation score, f(x) to maximize

Work overview

- Programming language:
  - Python
- Work developed:
  - City and schedule file parsing
  - Solution evaluation via path simulation
  - Multi-processing GA
  - Alternative ILS, SA and TS
  - Pygame view of the roads

- Development Environment:
  - VSCode
  - MVC architectural pattern
  - Test-driven development
- Python Data Structures:
  - Deque, List, Dictionary, Set



The approach: Evaluation function

- The evaluation function, f(city,solution), returns the simulation score of the schedule solution, in the given city, which we are trying to maximize. We managed to obtain the same score as Google intended, following strict rules:
  - Only one car can go through a green light in a second
  - All cars start at the end of the initial street of their path, waiting for the green light
  - Cars pile up at the end of a street in a first-in, first-out manner
  - Cars must reach the end of the final street in their path to score
  - Bonus points are given to each car for each second they are done and the simulation is up
  - The green light schedule for each intersection repeats in a cycle fashion during the simulation

The approach: Genetic Algorithm

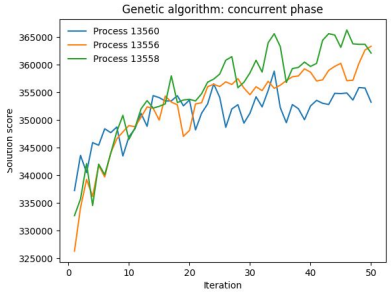
- The genetic algorithm is comprised of two phases.
  - The first phase subdivides the initial population into subgroups, which evolve in parallel. These subgroups not only aim to up their score, but also to be diverse - a bonus is given to schedules whose chromosomes are rarer, regardless of how well they perform.
  - The second phase merges back again the subgroups and has them evolve as a single population.
- Each generation takes the top ¼ of the population and breeds them with another random schedule. The crossover is made at the intersection level, meaning that the children will copy the intersection schedule from one of their parents for each intersection.
- The two-phased approach allows the second phase of the genetic algorithm to start off with a fitter and a lot more diverse population than a regular single-phase version. This enabled us to reach higher scores faster while keeping the algorithm with a reasonable execution time.

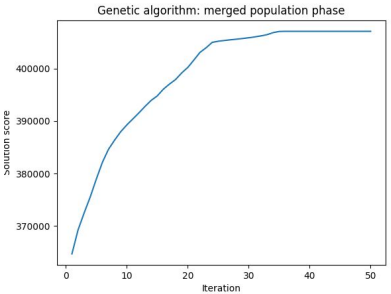
The approach: ILS, TS and SA

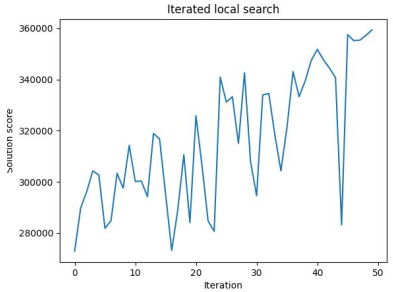
- Iterated Local Search
  - Uses mix with new solution perturbation operator; decrease strength as time passes
  - Proved to be a fast and efficient algorithm
- Taboo Search
  - Taboo criterion based on mutated intersection
  - Weak criterion in big cities with lots of intersections
  - Hard to find a best criterion (no concept of “group of adjacent intersections”)
- Simulated Annealing
  - First phase using mix with new solution neighbourhood operator
  - Second phase using intersection mutation operator
  - Third phase using single street mutation operator
  - Simple schedule logarithmic function
  - Simple and reliable algorithm

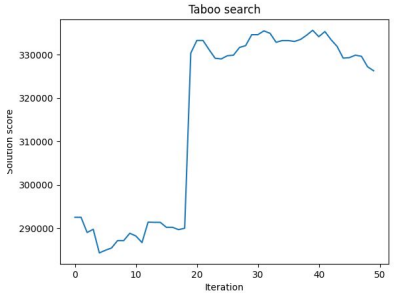
Experimental results (for City E)

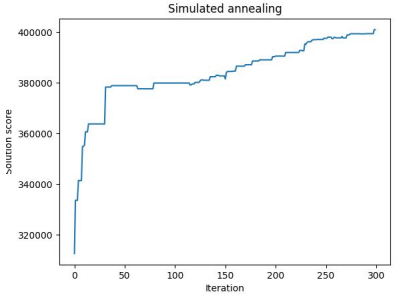
Multi-process genetic algorithm pushed the initial score higher than all other algorithms.











Conclusion

During the course of our work, several approaches were implemented and tested. The drive for a better result led us to explore different heuristics, algorithms and parameters, as well as out of the box neighbourhood definitions. Although the results were slightly distanced from the optimal ones, the improvements shown by our algorithms are still very notorious.

We believe this project to have been a positive experience, as it effectively demonstrated us the creative process behind basic artificial intelligence projects, prompting us to more deeply explore the approached topics.