IART- PROJECT 1

Topic 3: Metaheuristics for Optimization/Decision Problems

Traffic Signaling

Sofia Germer up201907461

Sérgio Estévão up201905680

Pedro Silva up201907523

Specification of the work to be performed

- **Task**: Given the description of a city plan and planned paths for all cars in that city, optimize the schedule of traffic lights to minimize the total amount of time spent in traffic and help as many cars as possible reach their destination before a given deadline.
- **Problem Description:**
  - **City Plan**
  - **Traffic Lights**
  - **Cars**
  - **Input format**
  - **Output Format**

`6 4 5 2 1000`	The simulation lasts **6** seconds, there are **4** intersections, **5** streets, and **2** cars; and a car scores **1000** points for reaching the destination on time.
`2 0 rue-de-londres 1`	Street **rue-de-londres** starts at intersection **2**, ends at **0**, and it takes L=**1** seconds to go from the beginning to the end.
`0 1 rue-d-amsterdam 1`	Street **rue-d-amsterdam** starts at intersection **0**, ends at **1** and has L=**1**.
`3 1 rue-d-athenes 1`	Street **rue-d-athenes** starts at intersection **3**, ends at **1** and has L=**1**.
`2 3 rue-de-rome 2`	Street **rue-de-rome** starts at intersection **2**, ends at **3** and has L=**2**.
`1 2 rue-de-moscou 3`	Street **rue-de-moscou** starts at intersection **1**, ends at **2**, and has L=**3**.
`4 rue-de-londres rue-d-amsterdam rue-de-moscou rue-de-rome`	The first car starts at the end of **rue-de-londres** and then follows the given path.
`3 rue-d-athenes rue-de-moscou rue-de-londres`	The second car starts at the end of **rue-d-athenes** and then follows the given path.

Figure 1 : Example of input file

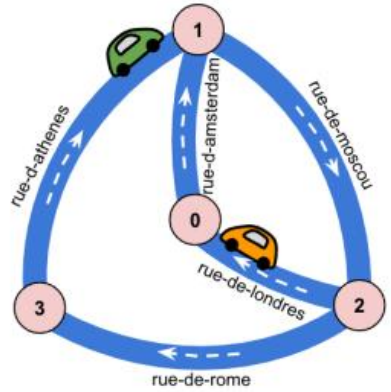

Figure 2 : The streets and intersections, as given by the example input data set, as well as the two cars at their initial positions.

You can check all the information about this problem in this link: https://storage.googleapis.com/coding-competitions.appspot.com/HC/2021/hashcode_2021_online_qualification_round.pdf

Formulation of the problem as an optimization problem

- **Solution representation**
  - The solution describes the traffic light schedule for specific intersections (an array with the green light duration of each traffic light and its place in the order of green light transition of the belonging intersection)
- **Neighborhood/mutation**
  - Swap the greenlight duration between two traffic lights;
  - Increment to a random traffic light a random duration;
  - Swap two traffic lights in the green light transition order of an intersection
- **Crossover functions**
  - Swap segments from two parent chromossomes with one point crossover.
- **Hard constraints**
  - **City:** Each street does not contain any intersection in between (if 2 streets need to cross outside an intersection, a bridge or tunnel is used);
  - **Traffic lights:** Each street can appear at most once in the schedule.
  - **Cars:** The path a car is going to drive through are defined by the input datasets and can't be altered
  - There will never be 2 streets connecting two intersections in the same direction.
  - **Submission file**: Each intersection can only be listed once in the submission file and each street can only be listed once per schedule.
- **Evaluation functions**
  - A score is awarded for each car that finishes its path before the end of simulation. If a car finishes at time T it scores:
    - F + (D-T) points if T <= D , where F is the fixed award for finishing the path and (D-T) means that it gets one point for each second left when the car finished the path
    - 0 points otherwise

```
3                   There are 3 intersections with traffic light schedules:
1                   On intersection 1 the traffic lights are green for
2                   2 different incoming streets, namely
rue-d-athenes 2     rue-d-athenes for 2 seconds, then green for
rue-d-amsterdam 1   rue-d-amsterdam for 1 second, then again rue-d-athenes.
0                   On intersection 0 the traffic lights are green for
1                   1 incoming street only, namely
rue-de-londres 2    rue-de-londres for 2 seconds per cycle (it's always green
                    for rue-de-londres).
2                   On intersection 2 the traffic lights are green for
1                   1 incoming street only, namely
rue-de-moscou 1     rue-de-moscou for 1 second per cycle (it's always green for
                    rue-de-moscou).
```

Figure 1: Example of submission file

Problem visualization

Intersection1				Intersection2			
Street1		Street2		Street1		Street2	
Street-name	3	Street-name	1	Street-name	0	Street-name	5

Table 1 : Problem representation

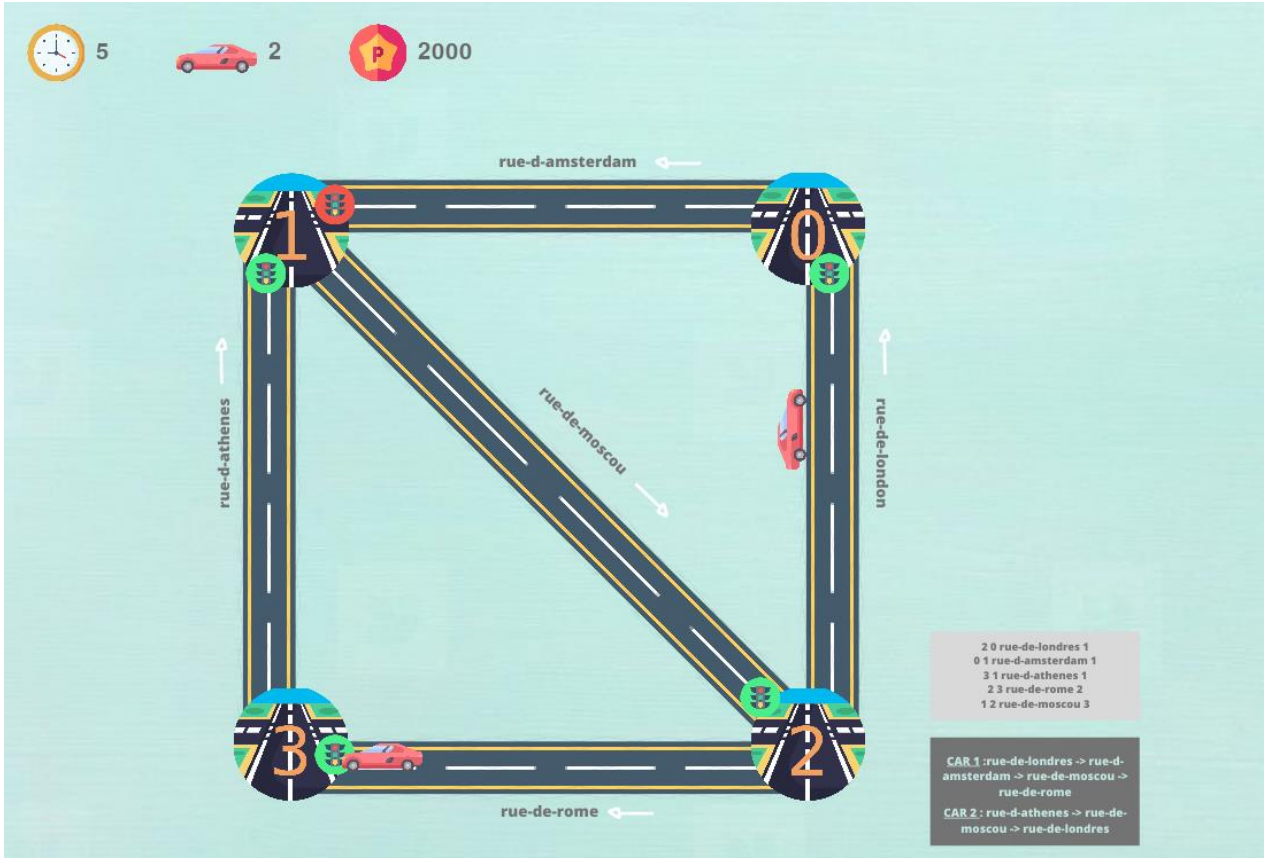

Figure 2 : Solution visualization developed in pygame

Developed Approach

- **Heuristics:**
  - **Random heuristic:** attributes random traffic light durations to each street, used to provide an initial population to the genetic algorithm
  - **Greedy heuristic:** attributes traffic light durations to each street based on its flow of cars, and then for each intersection reduces the traffic light durations of each incoming street based on their greatest common divisor.

- **Evaluation Functions:** solutions are ranked based on their score calculated using the number of cars that achieved their destination and how early they got there before the end of the simulation
- **Operators:** Swap, Re-order and Increment of traffic light duration

Implemented Algorithms

Metaheuritic	Description	Parameters
Hill-Climb	Finds a random neighbor of the current best solution using a mutation function. Keeps the neighbor if it's better than the best solution.	Number of iterations, probability of chosen mutation, light variation amplitude.
Simulated Annealing	Finds a random neighbour solution of the current best solution using a mutation function. Evaluates the neighbour solution and keeps it if it's better than the current best solution or, if it is worse, keeps or discards it based on the current algorithm's temperature.	Starting temperature, cooling schedule, number of iterations, minimum temperature, number of iterations per temperature, probability of chosen mutation, light variation amplitude.
Genetic Algorithm	Generate an initial population randomly or seeding with some previous generated solution. Evolve the population preforming crossover and mutations on certain individuals. Of the new generation, select the surviving chromosomes based on a roulette, with the probability proportional to each chromosome's score.	Initial population, size of population, number of generations, size of crossover, probability of crossover, elitism probability, probability of mutation, probability of chosen mutation.
Tabu Search	Finds a group of random neighbour solutions of the current best candidate solution using a mutation function. Evaluates the neighbour solutions and keeps the best one as the best candidate solution or, if it is worse, keeps or discards it based on the Aspiration criterea. The new best candidate solution is added to the tabu list. If best candidate solution is better than the best solution, it becomes the best solution as well.	Neighbor solutions list size, tabu list size, number of iterations, Aspiration Criterea, probability of chosen mutation, light variation amplitude.
Iterative Local Search	Finds a random neighbor of the current best solution using a mutation function and performs a Hill-Climb algorithm on it. Keeps the the result of the Hill-Climb if it's better than the best solution.	Number of iterations, probability of chosen mutation, light variation amplitude.

Experimental Results

Heuristics				
Input File	Random (max light duration = 3)		Greedy	
	Time(s)	Points	Time(s)	Points
a.txt	5.984e-05	1003	5.531e-05	2002
b.txt	0.03	244948	0.072	4525383
e.txt	0.003	236244	0.002	645058
f.txt	0.021	59014	0.015	476649

Hill-Climb (e.txt, starting solution: random (260837p), probability of chosen mutation: 50%, light var: simulation's duration)		
Num Iterations	Time(s)	Points
250	138.536	268938
500	280.764	267043
750	442.987	267598

Simulated Annealing (e.txt, max iterations: 500, starting solution: greedy, starting temp: 3000, runs per temperature: 1, probability of chosen mutation: 50%, light var: 3)			
Cooling function	Min Temperature	Time(s)	Points
exponential	0.01	71,906	643065
	0.005	77,969	644076
	0.001	89,595	645074
logarithmic	0.01	301,107	639420
	0.005	304,799	644076
	0.001	308,432	644314
linear	0.01	300,821	648289
	0.005	303,092	645571
	0.001	305,432	646702
quadratic	0.01	296,960	649367
	0.005	295,915	649020
	0.001	301,004	650772

Experimental Results - Continuation

Tabu Search
(e.txt, max iterations: 100, starting solution: greedy, probability of chosen mutation: 50%, light var: 3)

Neighbor Size	Tabu List Size	Time(s)	Points
5	10	295,662	647214
	20	303,465	650525
10	10	594,680	652061
	20	600,544	652976

Iterative Local Search
(e.txt, starting solution: greedy, probability of chosen mutation: 50%, light var: 3)

Num iterations	Time(s)	Points
4	127,470	645058
6	192,731	645058
8	250.362	645058
10	324.299	645058

Genetic Algorithm
(e.txt, starting solution: random, probability of chosen mutation: 50%, light var: simulation's duration)

Population Size	Elite	Nº iterations	Mutation probability (%)	Mutation ratio	Time (s)	Points
25	5	30	20	0.1	200	283022
50	10				406	296651
100	20				810	296839
50	10	50	20	0.1	667	302192
		70			916	291289
		90			1217	306590
50	10	50	10	0.1	988	295028
			30		641	300952
			40		657	292941
50	10	50	40	0.01	649	303359
				0.2	684	291898
				0.3	696	287033
100	20	100	0.3	0.01	3108	321743

Conclusion

- To solve the Traffic Signaling problem, we implemented heuristics and metaheuristics to find and optimal solution.
- Using an array to represent the solution of the problem facilitated the application of mutations and other function to the solution.
- Applying constraints on the random heuristic and neighbor function produced better solutions more consistently, excluding solution worst solution and limiting their quality.
- Using a wide variety of optimization algorithms made possible to evaluate their effectiveness in terms of this specific problem.
- This project allowed us to get an understanding about optimization algorithms and heuristics as well as how they impact and help solve real-world problems

Bibliography

- https://github.com/sbrodehl/hashcode2021/tree/main/Online%20Qualifications
- http://rbanchs.com/documents/THFEL_PR15.pdf
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.648.2309&rep=rep1&type=pdf
- https://sboyles.github.io/teaching/ce377k/heuristics.pdf