Lab assignment 2: *Constraint Satisfaction and Heuristic Search*
Heuristics and Optimization
Computer Science, 2020–2021

1. Goal

The goal of this assignment is to learn to model and solve constraint satisfaction and heuristic search tasks.

2. Problem statement

An important space agency has contacted the students of *Heuristics and Optimization*. After a number of meetings you and your partner have been selected to solve various of their problems.

2.1. Part 1: Assignment of transmission antennae to satellites

The number of satellites orbiting around the earth has grown significantly during the recent years and thus, the number of requests to assign transmission antennae on earth to different time slots has increased significantly as well. The space agency has reported they have six satellites and twelve transmission antennae. Each satellite is visible only during one or two time slots by one or more antennae. Table 1 shows the relationship between satellites, time slots and antennae. The first column shows the satellite identifier, the second column shows a time slot under which the satellite is reachable for those antennae shown in the third column.

The space agency is interested in determining an assignment of satellites to time slots and antennae considering the following constraints:

1. All satellites must have a transmission antenna assigned to it in all its time slots. Take SAT3 for instance with two visible time slots (6:00–12:00 and 13:00–16:00), so that an antenna has to be assigned to its first time slot and another one for the second, among those with visibility in each time slot.

2. Since SAT1 and SAT2 have similar orbits, it is required to assign them the same antenna

3. Satellites SAT2, SAT4 and SAT5 should have assigned different antennae.

4. In case SAT5 communicates with ANT12, then SAT4 can not communicate with ANT11.

5. If, in any solution, ANT7 and ANT12 are used, then both must be assigned to time slots beginning before 12:00 or after. It is not allowed, for example, for ANT7 to be assigned the time slot 06:00–12:00 to SAT3, and ANT12 to the time slot 16:00–00:00 for communicating with SAT4.

Satellite	Time slot	Antennae
SAT1	00:00 - 12:00	ANT1, ANT2, ANT3, ANT4
SAT2	00:00 - 12:00	ANT1, ANT2, ANT3
SAT3	06:00 - 12:00	ANT4, ANT6
SAT3	13:00 - 16:00	ANT7, ANT9, ANT10
SAT4	16:00 - 00:00	ANT8, ANT11, ANT12
SAT5	06:00 - 13:00	ANT1, ANT7, ANT12
SAT6	09:00 - 13:00	ANT7, ANT9
SAT6	13:00 - 19:00	ANT3, ANT4, ANT5

Tabla 1: Satellites, time slots and atennae

It is required to:

- Model this problem as a *Constraint Satisfaction Processing* (CSP) task.
- Use the python library `python-constraint` to implement the previous model and determine the antennae and time slots to assign to each satellite.

  The program has to be executable from a console with the following command:

  ```
  $ python CSPScheduling.py
  ```

  Each team should generate different test cases, e.g., modifying the different constraints and adding/-removing satellites/antennae, widening or narrowing the time slots, with the purpose of analyzing the behaviour of the program.

2.2. Parte 2: Heuristic Search

The same space agency requires us to solve a diferent problem. Figure 1 show satellites SAT1 and SAT2 orbiting around the earth using different trajectories, each with a different visibility band.

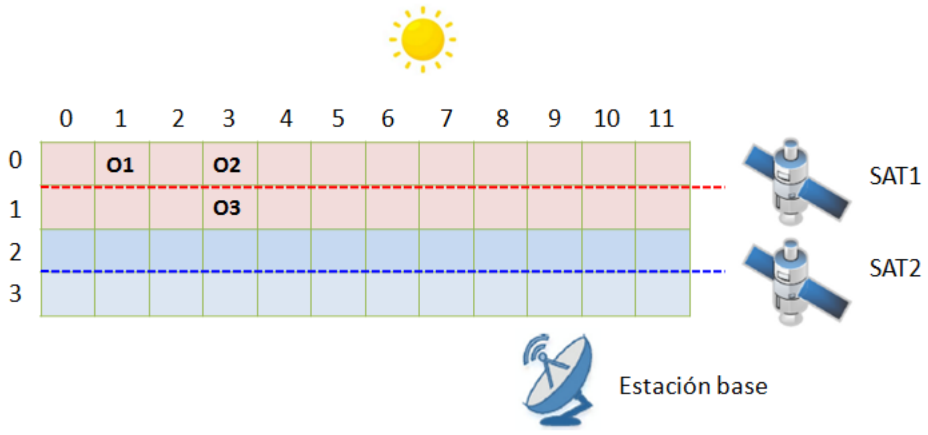


Figura 1: An example with two satellites and three observation points

So far, satellite SAT1 has attached the visibility bands 0 and 1 in Figure 1, whereas SAT2 is associated with the visibility bands shown in rows 2 and 3. Each satellite can take measurements only of those objects that are reachable in its visibility band. In addition, satellites are visible only 12 hours a day so that they are out of reach during the other half of the day. During their visibility period, satellites can perform various activities any of them spanning over a period of exactly one hour: either they can do nothing; or they can take a measurement of the terrain behind them; or they can downlink their data to the base station; or they can turn to point to a different visibility band; or they can perform battery charging operations using their solar panels. Of course, satellites can perform only one of these activities and it is not allowed to perform more simultaneously. When turning to a different visibility band, SAT1 can move from its visibility band $(0, 1)$ to $(1, 2)$ and vice versa, and SAT2 can change from $(2, 3)$ to $(1, 2)$ and vice versa. Each satellite is equipped with a battery with a specific number of units of energy and the number of energy units consumed when taking measurements, downlinking data or turning to a different visibility band are known. Besides, each battery charging operation increases the battery life in a given number of energy units.

As an example, consider Figure 1. It shows three different observations on earth located in diferent points: $O1$, $O2$ and $O3$. It can be safely assumed that both SAT1 and SAT2 take one energy unit for the operations of taking measurements, downlinking data or turning to a different visibility band. However, SAT1 has a battery with capacity for only 1 energy unit, whereas the capacity of the battery of SAT2 is equal to 8 units. The resulting plan that allows taking the three observations and to downlink them to the base station is shown next:

```
1. SAT1: IDLE, SAT2: IDLE
2. SAT1: Measure O1, SAT2: IDLE
3. SAT1: Charge, SAT2: Turn
4. SAT1: Measure O2, SAT2: Measure O3
5. SAT1: Charge, SAT2: Donwlink O3
6. SAT1: Downlink O1, SAT2: IDLE
7. SAT1: Charge, SAT2: IDLE
8. SAT1: Downlink O2, SAT2: IDLE
```

Each line in the preceding sequence shows the identifier of a satellite and the action to perform. For instance, the first line states that during the first hour both satellites do nothing; in the second hour, SAT1 takes a measurement of $O1$, while SAT2 does nothing. As a matter of fact, only SAT1 can take a measurement of $O1$ since this object is unreachable for SAT2 even if it would turn to a different visibility band. When making the measurement, SAT1 consumes one unit of energy so that it needs to charge its battery during the following hour to continue operating, while SAT2 turns to the visibility band $(1, 2)$ to gain access to object $O3$. In the next hour, SAT1 takes a measurement of $O2$ and SAT2 does the same with $O3$. Finally, both observations are downlinked to the base station. Note it is not necessary at all to take/downlink data in any specific order. The plan is successful once all programmed measurements have been sent back to the base station.

Note that it might not be possible to make all measurements in just one orbit of the satellites through their visibility band, so that the satellites should then continue orbiting during the half of the day when they have no access to the base station before resuming their operations. During this part, the only allowed operation is to do nothing which actually consumes no energy. As an example of this case, note that if a measurement $O4$ is programmed to take place in position $(0, 4)$, only SAT1 could reach it, but SAT1 can not take this measurement in the same orbit it takes the measurement $O2$, because after making the measurement of $O2$ it has necessarily to charge its battery before operating again. Thus, it will be during the next orbit when SAT1 could make the measurement of $O4$ and to downlink data to the station base.

It is required to:

- Model this problem as a *Heuristic search* task.
- Implement A* (in any programming language) that solves the problem optimally with two *admissible* and *informed* heuristic functions that estimate the effort to reach the goal.

  The resulting code should be executable from a console with the following command:

  ```
  $ ./Cosmos.sh <problem.prob> <heuristic>
  ```

where:

- `Cosmos.sh`: name of the script that invokes the program submitted to this lab assignment
- `problema.prob`: Filename that contains the instance to solve. The contents of the file modelling the case shown in Figure 1 are shown next:

  ```
  OBS: (0,1);(0,3);(1,3)
  SAT1: 1;1;1;1;1
  SAT2: 1;1;1;1;8
  ```

  where the first line shows the coordinates of all the programmed observations; the next two show the configuration of each satellite which consists each of a semi-colon ('`;`') separated list with the following fields: the number of energy units consumed for each measurement, donwlink, turn to a different visibility band, the number of units that are gained with every battery charging operation, and the number of energy units initially available in the on-board battery.

- `heuristic`: name of the heuristic to use for solving the specific instance given in `problem.prob`. The different values for this argument should be detailed in the report and should be also available in the online help.

The program should generate two different output files in the same directory where the input file is located and should be named after the name of the problem plus an additional extension:

- Solution file. It should contain the operations that should be performed by each satellite in order to make all the programmed measurements and to donwlink them to the base station. An example of the output format was shown above when discussing a feasible solution to the case depicted in Figure 1. This file should be suffixed with '`.output`'

  Example: `problema.prob.output`

- Statistics file. This file should contain various data about the search process such as the overall running time, overall cost, step length, number of expansions, etc. An example is shown below:

  ```
  Overall time: 145
  Overall cost: 54
  # Steps: 27
  # Expansions: 132
  ```

  The name of this file should be suffixed with `.statistics`.

  Example: `problem.prob.statistics`

- Propose different test cases with different configurations (different number of programmed measurments or a different satellite configurations) and solve them using the program submitted. These test cases should be devised according to the efficiency reached by the proposed implementation.

- Perform a comparative analysis of both heuristics —e.g., comparing the number of expansions, overall running time, etc.

3. Requirements for the report

The report must be delivered in PDF format and it should consist of a maximum of 15 pages, including the cover, the table of contents and the back cover. It should contain, at least:

1. Brief introduction explaining the contents of the document.
2. Description of the models, discussing the decisions carried out.
3. Analysis of results.
4. Conclusions.

Important: Reports delivered in a format other than pdf will be penalized with 1 point.

The report **should not include source code** in any case.

4. Grading

This lab assignment will be graded over 10 points. To assure that the assignment is graded you must do at least the first part and the report.

The distribution of points will be as follows:

1. Part 1 (3 points)
   - CSP model 0.75 points)
   - Implementation of the model (1.25 points)
   - Resolution and analysis of different test cases (1 point)
2. Part 2 (7 points)
   - Problem Modeling (1 point)
   - Model Implementation (3 points)
   - Resolution and analysis of different test cases along with a comparative analysis (3 points)

When grading the model proposed, a correct model will result in half of the points. To obtain the whole score, the model must:

- Be correctly formalized in the report.
- Be simple and concise.
- Be properly explained (it should remain clear what is the reason for each component of the model).
- Justify in the report all the design decisions taken.

When grading the model implementation, a correct implementation will result in half of the points. The implementation provided must compile without trouble and should implement the specifications described herein. To obtain the whole score, the implementation must:

- Faithfully implement the model proposed above.
- Deliver source code correctly organized and commented. Names shall be descriptive.
- Contain a good deal of different test cases that prove the vaidity of your implementation.

When grading the results analysis, it will be positively valued to include in the report personal conclusions about the assignment difficulty and about what you learnt while carrying it out.

5. Submission

The deadline for submitting the assignment is December, 22 at 23:55. This is a hard deadline and it will not be extended under any circumstances.

Only one student from each team must submit:

- A single `.zip` file to the assignment section of 'Aula Global' through the submission entry point named "*Second lab assignment*".

  This file must be named `p2-NIA1-NIA2.zip`, where `NIA1` and `NIA2` are the last 6 digits of each student's NIA padding with `0`s if necessary.

  Example: `p2-054000-671342.zip`.

- The report in pdf format shall be submitted through the Turnitin submission entry point enabled in Aula Global named "*Second lab assignment (only pdf)*".

  The report in pdf format and shall be named `NIA1-NIA2.pdf` —after properly substituting the NIA of each student.

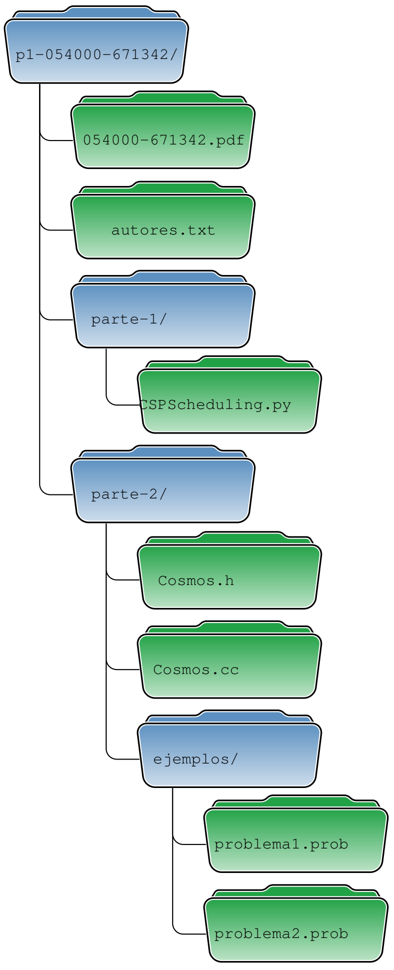
  Example: `054000-671342.pdf`.

Uncompressing the `.zip` file submitted through the first entry point must generate a directory named `p2-NIA1-NIA2`, where `NIA1` and `NIA2` are the last 6 digits of each student's NIA padding with `0`s if necessary. This directory shall contain: first, the same pdf submitted through Turnitin, and shall be named `NIA1-NIA2.pdf` —after properly substituting the NIA of each student; second, a file named `autores.txt` that identifies the members of them team, one per line with the format: `NIA Surname, Name`. For example:

```
054000 Von Neumann, John
671342 Turing, Alan
```

In addition, this directory shall contain also the directories called "`parte-1`" and "`parte-2`". The directories shall contain the necessary files for properly executing each part.

The following figure shows a plausible distribution of files after decompressing the `.zip` file:



Important: ignoring these guidelines one way or another might result in a loss of up to 1 point in the final score.