DEEC
DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Instituto Superior Técnico

Artificial Intelligence and Decision Systems (IASD)
Lab classes, 2015/2016

Assignment #1

Version 1.1 - October 01, 2015

ISTravel

1 Introduction

The goal of this project is to develop a "travel agent" program. This program, written in Python and using Search-based algorithms, should determine the best route to take a client from an initial city to the final city.

All relevant cities are defined in a network of connections (a graph). Besides identifying the nodes (cities) and the edges (connections), this network also defines the types of transports there exist for each connection, and the corresponding travel duration and cost.

The program to be developed should produce, for each client, the best sequence of connections — the route — (and corresponding type of transport) that would allow the client to go from the initial city to the final one, according to a specified criterion: time or cost.

2 ISTravel - specification

In this project, it is considered the following four types of transport: bus, train, airplane and boat.

Figure 1 shows an example of a network (map) with all available connections (type of transport) between cities. In that map, between Lisboa and Santarém there is only one connection by bus, whereas between Faro and Beja there are two connections, one by train and another by bus.

Notice that, for instance, between Porto and Madrid there are several possible routes: a direct connection by train or bus, a route through Chaves (train or bus), a route through Aveiro, Faro, Lisboa, and so on... Choosing one of these possible routes depends on the client preference: if he intends to minimize time or cost.

In this project, the standard time unit is the minute, and a day has 1440 minutes. Moreover, connections have a predefined daily periodicity.

Therefore, when defining a connection, besides type of transport, time and cost, the following parameters are also defined:

- t_i - the time instant when the first transport departs, on each day - a value between 0 and 1440;

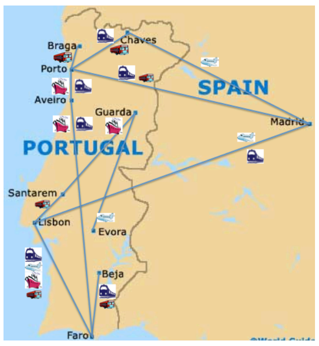

Figure 1: Example of a map for the travel agent.

- t_f - the time instant after which no transport departs, on each day - a value between 0 and 1440 and $t_f \geq t_i$;
- p - the daily periodicity.

Also, you should consider that connections are bidirectional, i.e., if there is a connection between A and B, there is also the same connection between B and A, with exactly the same parameters.

For instance, suppose there is a connection between city A and city B, with the following parameters: $t_i = 0$, $t_f = 1010$ e $p = 200$. This means that there are transports leaving A towards B (and vice-versa) at time instants: $t = 0$, $t = 200$, $t = 400$, $t = 600$, $t = 800$, e $t = 1000$, for the first day, $t = 1440 + 0$, $t = 1440 + 200$, $t = 1440 + 400$, $t = 1440 + 600$, $t = 1440 + 800$, e $t = 1440 + 1000$, for the second day, and so on... It is assumed that the transportation fleet is infinite, so there is always a transport available to depart on time from any city.

Another example of a connection parameterization is the following: $t_i = 100$, $t_f = 1010$ e $p = 1000$. In this case, there is only one transport departing at time instant 100. So, whenever $p > t_f - t_i$, the connection has only one daily transport departing at t_i. Furthermore, in this case, the next transport will depart next day at time $1540 (= 1440 + 100)$

Additionally, clients may have some constraints on the acceptable connections. In this project, you should consider the following **type A** constraints, but only one type A per client:

A.1 defines a particular type of transport that the client **don't want to use**;

A.2 the maximum time for a connection;

A.3 the maximum cost for a connection.

For instance, suppose that a client establishes (through a A1 constraint) that he doesn't want to travel by airplane. This means that the program should determine a route where none of the connections uses airplane. But if the client defines (through a A2 constraint) that the maximum time per connection is 1000, then the route might have a total time greater than 1000, but each connection of it must have a duration smaller than or equal to 1000. The same happens in terms of cost.

Finally, still for each client, he may define an additional constraint (**type B**), but only one type B per client:

B.1 if time is the variable to optimize, the client may establish that he doesn't accept routes with a total time greater than a specified value; or

B.2 if cost is the variable to optimize, the client may establish that he doesn't accept routes with a total cost greater than a specified value.

3 ISTravel - program

The program to be developed should be able to read the configuration of a network of cities and requests from the clients, and for each client it should determine the route that serves best the client.

Of course, the program should be prepared to handle impossible requests, and it should not assume any resemblance between the input data and reality.

In the following sections, it is presented the way the program should be called, the structure of the input files and the format for the output file.

3.1 Program execution

The program ISTravel should be executed in the following way:

```
$ python istravel.py input1.map input2.cli
```

where:

`istravel.py` is the python file where execution starts;

`input1.map` is an input file where the network is defined, according with the format presented in Section 3.2.1.

`input2.cli` is an input file where all clients' requests are defined, according with the format presented in Section 3.2.2.

The input files always have extensions `.map` for the network and `.cli` for the clients, but may have any names. Your program may assume that the input files are always in the correct format.

3.2 Input files format

The input information for the ISTravel program is given in two input files.

3.2.1 File .map

The configuration of the network of cities is defined in a file with extension `.map` obeying the following rules:

- the first row contains two integer numbers, N e L, where N represents the number of cities and L the number of connections.
- the following L rows correspond to the existing connections. Each row (connection) contains eight (8) elements separated by one blank space:
  - two integer numbers identifying the two cities connected;
  - the type of transport, one of the following words[1]: **aviao**, **comboio**, **barco** or **autocarro**;
  - time duration;
  - cost;
  - time instant of the first depart on each day (t_i);
  - time instant after which there are no more departures (t_f); and
  - daily periodicity (p).

  In order to identify the cities, assume that they are numbered from 1 to N.

Note: time and costs are always represented by integer numbers.

As an example, the file that describes the network[2] presented in Figure 1, is shown in Figure 2. In the example, the cities are numbered as follows: 1 - Lisboa, 2 - Évora, 3 - Beja, 4 - Faro, 5 - Santarém, 6 - Guarda, 7 - Madrid, 8 - Aveiro, 9 - Porto, 10 - Braga, 11 - Chaves, 12 - Dili.

3.2.2 File .cli

The description of clients' requests is made in a file with extension `.cli`, obeying the following rules:

- first row contains an integer number, C, representing the number of clients.
- the following C rows correspond to the requests of the clients. Each row (client) contains six (6) mandatory elements separated by one blank space:
  - an integer number identifying the client number (a number between 1 and C);
  - two integer numbers identifying the initial and final cities;
  - a non-negative integer number representing the time instant after which the client is available to travel;

[1]Portuguese words are used here because the input files were defined using those words.
[2]with a new city - Dili, without connections.

```
12 21
4 3 comboio 60 10 0 1440 100
4 1 comboio 100 14 500 1200 500
4 3 autocarro 70 9 600 1440 300
1 4 aviao 10 30 480 1440 1440
4 1 barco 90 21 0 1000 200
4 1 autocarro 90 15 0 1000 200
1 5 autocarro 20 5 0 1000 200
2 6 aviao 80 14 0 1000 200
7 1 aviao 200 100 0 1000 200
1 7 comboio 1111 75 0 1000 200
4 8 barco 220 93 0 1000 200
4 8 comboio 132 61 0 1000 200
9 8 barco 22 9 0 1000 200
9 8 comboio 13 13 0 1000 200
9 10 autocarro 5 2 0 1000 200
11 9 comboio 19 9 0 1000 200
11 9 autocarro 18 8 0 1000 200
9 7 comboio 302 101 0 1000 200
7 9 autocarro 1221 90 0 1000 200
11 7 aviao 2001 110 0 1000 200
5 6 barco 5 5 0 1000 200
```

Figure 2: File .map for the network presented in Figure 1.

  – the optimization criterion, identified by one the following words: **tempo** or **custo**. The first word means that the client wants a route that minimizes time, whereas the second wants to minimize cost; and
  – the number of constraints (0, 1 or 2).

Besides the mandatory elements, each row may also have one or two constraints:

  – a pair of two elements, where the first can be **A1**, **A2** or **A3**, and the second can be aviao, comboio, autocarro or barco if the first is **A1** or an integer number if the first is either **A2** or **A3**;
  – a pair of two more elements composed by one of the words **B1** or **B2**, followed by an integer number.

As an example, Figure 3 shows the information for a set of 5 clients.

In the example presented in Figure 3, each row has the following interpretation:

- client 1 wants to travel from city 1 to city 4, as fast as possible (time), is available to start at tine instant 0, and doesn't want to travel by airplane,

```
5
1 1 4 0 tempo 1 A1 aviao
2 2 3 250 custo 1 B2 10000
3 1 12 1550 custo 0
4 9 10 0 tempo 2 A2 500 B1 1440
5 5 9 1000 tempo 2 A3 200 B1 2000
```

Figure 3: Example of a file .cli.

- client 2 wants to travel from city 2 to city 3, as cheap as possible (cost), is available at time instant 250, and doesn't accept routes whose total cost is greater than 10000;
- client 3 wants to travel from city 1 to city 12, as cheap as possible (cost), is available at time instant 1550, and has no constraint;
- client 4 wants to travel from city 9 to city 10, as fast as possible (time), is available at time instant 0, and doesn't accept connections with a duration greater than 500 neither routes which total time exceeds 1440;
- client 5 wants to travel from city 5 to city 9, as fast as possible (time), is available at time instant 1000, and doesn't accept connections with a cost greater than 200 neither routes which total time exceeds 2000.

3.3 Output file format

The result of the ISTravel program execution consists of determining for each client the route that minimizes his preference, time or cost. The solution should saved in a output file with the same name as the `.cli` input file, but with the extension `.sol`. For instance, if the client file is named `input231.cli`, the output file should be named `input231.sol`.

The `.sol` file should have C rows, one for each client, with the following information, separated by a blank space:

- the client number, the determined route, the total time and the total cost. The total time is the difference between the arrival time instant to the final city and the time instant after which the client is available to travel (see definition in the .cli file). The total cost is just the sum of the costs of all connections used in the route.
- the route specification mentioned above should follow the following rules: starts with the number of the initial city, and ends with the number of the final city. In between, you must insert for each connection of the route two elements: the type of transport (aviao, comboio, barco or autocarro) and the number of the next city, all separated by a blank space.

For instance, if a route between city 33 and city 18 involves the following connections: train between city 33 and city 67, followed by airplane between 67 and 101, and boat between 101 and 18, the corresponding row in the output file should be[3]:

```
12 33 comboio 67 aviao 101 barco 18 1267 8197
```

- if there is no possible route for a particular client, the corresponding row should have only the client number and -1.

For the network presented in Figure 1, the output file could be:

```
1 1 autocarro 4 90 15
2 2 aviao 6 barco 5 autocarro 1 comboio 4 autocarro 3 1020 47
3 -1
4 9 autocarro 10 5 2
5 5 autocarro 1 autocarro 4 comboio 8 comboio 9 853 94
```

Figure 4: Example of a output file for the network presented in Figure 1 and defined in Figure 2, and for the clients defined in Figure 3.

[3]The initial number 12 is the client number; the last two numbers are the total time and the total cost.

4 Assignment Goals

1. Develop a Python program for solving this type of travel agent problem using Search methods. The solver must include at least one uninformed search method, and one informed search method with an heuristic function.

   In your program the domain-independent part *should be* explicitly isolated from the domain-dependent one.

   Suggestion: Use the General Search to develop a generic search algorithm (domain-independent part) and then particularize it to the travel agent problem (domain-dependent part).

   Note 1: All code should be adequately commented.

   Note 2: Do not forget to identify the version of Python used.

2. The answers to the following points should be included in a short **technical** report, delivered together with the source code.

   (a) Describe how do you represent the problem state, the operator(s), the initial state, the goal state, and the path cost;

   (b) Identify and justify each uninformed and informed search algorithms implemented;

   (c) Describe how did you separate the domain-independent part of the solver from the domain-dependent one;

   (d) Describe and justify the heuristic function(s) developed;

   (e) Compare quantitatively and comment the performance of the different search techniques implemented.

The deliverable of this Lab Assignment consists of two components:

- the Python source code
- the short report (**pdf** format) with no more than 2 pages.

Deadline*: 24h00 of 17-Out-2015, by email to lmmc@isr.ist.utl.pt*
(Projects submitted after the deadline will be penalized.)