TÉCNICO LISBOA

Artificial Intelligence and Decision Systems (IASD)
Mini-projects, 2019/2020

Assignment #1

Version 1.0 (9-Oct-2019)

Airline Scheduling and Routing

1 Introduction

The airline industry is one of the most competitive ones in the world economy. For instance, the pressure of the low-cost airliners pushed the whole industry to pursue cost efficiency. One critical components of this efficiency is the scheduling and routing of the airplanes, given a set of legs to be flown by the company. This project addresses the problem of finding the daily schedule and routing of airplanes that maximizes the company profit, hereby called the Airline Scheduling And Routing (ASAR) problem.

2 Problem statement

Let $\mathcal{A}$ be a set of airports, where for each $a \in \mathcal{A}$, t_a^{open} and t_a^{close} are the opening and closing times[1] of flight operations. Let $\mathcal{P}$ be a set of airplanes, each one $p \in \mathcal{P}$ belonging to a class $c = \pi(p)$, where $c \in \mathcal{C}$ with $\mathcal{C}$ being the set of all aircraft classes. Each aircraft class $c \in \mathcal{C}$ is characterized by a rotation time[2], d_c^{rot}.

The set of legs to be covered is denoted by $\mathcal{L}$, where each leg $l \in \mathcal{L}$ is a tuple $l = (a_{dep}, a_{arr})$ of airports, $a_{dep}, a_{arr} \in \mathcal{A}$. Each such leg is characterized by a fixed flight duration d_l, and a profit $p_{l,c}$ for each aircraft class $c \in \mathcal{C}$.

A schedule S_p of an airplane $p \in \mathcal{P}$ is a list of tuples (t, a) where t is the departure time from airport $a \in \mathcal{A}$ to the next airport. A schedule is subject to two consistency constrains:

1. no two consecutive airports in a schedule are equal, and
2. the first and the last airport in a schedule must be equal (because the schedule is daily).

[1] All times are UTC, that is, coordinated universal time.

[2] The rotation time is the time between a consecutive arrival and departure of an airplane at an airport.

The starting airport of an airplane can be any airport. Note that the time between one arrival and the next departure of an airplane $p \in \mathcal{P}$ is given by its rotation time $d^{rot}_{\pi(p)}$. In addition, the time t of any arrival or departure in any airport $a \in \mathcal{A}$ must lie between the corresponding opening and closing times, that is $t^{open}_a \leq t \leq t^{close}_a$.

The problem consists in finding a set of schedules $\mathcal{S} = \{S_p\}$ that flyes all legs in $\mathcal{L}$ exactly once, and that minimizes the total profit $P(\mathcal{S})$ over all legs flown by all schedules, that is,

$$P(\mathcal{S}) = \sum_{S_p \in \mathcal{S}} \sum_{l \in S_p} p_{l,\pi(p)} \tag{1}$$

3 Objectives

The goal of this mini-project is to solve the problem described in the previous section, using search methods. This includes defining a state representation, the operators, and the goal condition, allowing an appropriate search method to find the optimal solution.

The implementation should be done in Python version 3. No extra modules, besides the Python Standard Library, are allowed. The search algorithm implementations are the ones from the GitHub repository of the course textbook[3], namely the module `search.py` available from `https://github.com/aimacode/aima-python`.

In particular, the problem should be implemented as a Python class with name `ASARProblem`, that derives from the abstract class `search.Problem`, and that defines (at least) the following methods:

`actions(`s`)` returns a list (or a generator) of operators applicable to state s

`result(`s`, `a`)` returns the state resulting from applying action a to state s

`goal_test(`s`)` returns `True` if state s is a goal state, and `False` otherwise

`path_cost(`s`)` returns the path cost of state s

`heuristic(`n`)` returns the heuristic of node[4] n

`load(`f`)` loads a problem from a (opened) file object f (see below for format specification)

`save(`f`, `s`)` saves a solution state s to a (opened) file object f (see below for format specification)

[3] Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Pearson Education Limited.

[4] In the search algorithms implementations, a node encapsulates a state. The state of a node `n` can be obtained using `n.state`.

3.1 Input file format

This file format specifies a ASAR problem. It is a text file, organized by lines. Empty lines should be ignored. Non-empty lines should have one of these forms:

`A <code> <start> <end>`

to specify an airport with code[5] `<code>`, and opening time `<start>` and closing time `<end>`. Both of these times are UTC with the format *hhmm*;

`C <class> <dr>`

to specify aircraft class `<class>` with rotation time d_c^{rot} given by `<dr>` in UTC with the format *hhmm*;

`P <airplane> <class>`

to specify an airplane, registration code `<airplane>`, belonging to class `<class>`; and

`L <dep> <arr> <dl> <class1> <pl1> <class2> <pl2> ...`

to specify a leg from airport `<dep>` to airport `<arr>`, with flight duration d_l given by `<dl>`, and the profits for all aircraft classes $\mathcal{C}$, given in pairs of aircraft class $c \in \mathcal{C}$ and corresponding profit $p_{lc} \in \mathbb{R}$: `<class1> <pl1>`, `<class2> <pl2>`, and so on.

3.2 Output file format

The solution to the ASAR problem should be returned using the following file format. Again, it is a text file, where blank lines are ignored, and each non-blank line should have one of these two formats:

`S <airplane> <time1> <dep1> <arr1> <time2> <dep2> <arr2> ...`

to specify a schedule of airplane code `<airplane>`, where each triple `<time`i`>` `<dep`i`>` `<arr`i`>`, for $i = 1, \ldots$, comprises a leg from airport `<dep`i`>` to `<arr`i`>`, with departure time `<time`i`>`; and

`P <profit>`

to provide `<profit>` as the total profit of the solution, given by (1).

If the input problem is infeasible, that is, there is no solution that satisfies the constraints, the file should contain a single line containing `Infeasible`.

[5] In the examples, the ICAO airport codes is used, but implementations should not rely on this fact.

4 Evaluation

The deliverable for this mini-project has two components:

- a single Python file, called `solution.py`, implementing the above mentioned `ASARProblem` class, and
- a report in the form of a short questionnaire.

Both components are submitted to a Moodle platform. Instructions for this platform are available at the course webpage.

The grade is computed in the following way:

- 30% from the public tests
- 30% from the private tests
- 30% from the questionnaire
- 10% from the code structure

Deadline: 9-Nov-2019 (Projects submitted after the deadline will not be considered for evaluation.)

A Example files

For the ASAR problem specified with file

```
A LPPT 0600 2300
A LPPR 0600 2200
A LPFR 0800 2000
A LPMA 0800 2200

P CS-TUA a330
P CS-TVA a320

L LPPT LPPR 0055 a320 100 a330 80
L LPPR LPPT 0055 a320 100 a330 80
L LPPT LPFR 0045 a320 80 a330 20
L LPFR LPPT 0045 a320 80 a330 20
L LPPT LPMA 0145 a320 90 a330 120
L LPMA LPPT 0145 a320 90 a330 120

C a320 0045
C a330 0120
```

a solution is given by the file

```
S CS-TUA 0800 LPMA LPPT 1105 LPPT LPMA
S CS-TVA 0800 LPFR LPPT 0930 LPPT LPPR 1110 LPPR LPPT 1250 LPPT LPFR
P 600.0
```