Airline staffing problem

Original lab session by Gérard Verfaillie.

This problem is about the staffing of an airline, i.e. assigning pilots and crew members to scheduled flights using a Constraint Satisfaction/Optimisation Problem (CSP/COP) solver (here `facile`). Three instances of the problem are given.

As a pure CSP modelling fails for the larger instance, we will solve it using a combination of CSP and Mixed-Integer Linear Programming modelling. You may refer to past classes with the `PuLP` library for basic usage.

Problem description

An airline operates a given set of flights between pairs of airports. Each flight must be staffed with two pilots and enough cabin crew members depending on the type of aircraft.

Each staff member must start its trip from his hometown (not before 12 am) and be back home in the evening (before 12 am). They must not fly more than a maximum number of flights per day, and more than a maximum amount of time out of home (from first take-off time to last landing time in addition to commuting time).

Each staff member may fly as a passenger if they just need to be transferred to another airport/city. They may as well be assigned to just stay home.

We will first search for a solution to this problem, then find the optimal solution that will maximise the money made by the airline by minimising the number of passenger seats assigned to staff.

Problem data

The following data are given:

- `Nv`: total number of flights;
- `Na`: total number of airports;
- `Ni`: total number of cities;
- `Va`: a table associating a city to an airport (they may be several airports in the same city);
- `Ov` (resp. `Dv`): this table associates an origin (resp. destination) airport to a flight;
- `Td` (resp. `Ta`): this table associates a take-off time (resp. landing time) to a flight, (in round hours);
- `Dt`: this table returns transit times between two airports of the same city (in hours). This time is equal to 25 (more than a day, so infinite) for

airports in different cities; `Dt[i][i]` is the time required to transfer between two aircraft at airport *i*;

- `Np`: this table associates a number of passenger seats to a flight;
- `Nec`: this table associates a number of required cabin crew members to a flight;
- `Pr`: this table associates a price for a ticket in a flight;
- `Ne`: total number of staff in the airline;
- `Ty`: this table associates 1 to a pilot, 0 to a cabin crew member;
- `Vh`: this tables associates a hometown to each staff member;
- `Nvmax`: maximum number of flights per staff member per day;
- `Dmax`: maximum work time (out of home) per staff member per day;
- `Dda`: standard commuting time (both ways).

Warning.

- Three instances are given (tiny, small, normal). Start debugging with the tiny instance; when it works, try the small instance: **you will be graded based on the results of the small instance**.
- Keep the normal instance for next section. (Trying it with this modelling may stall the kernel and require a restart)

Index 0 in the tables printed below means “no flight”, associated to “no airport” and “no city”.

```
from airlines import tiny, small, normal
```

```
>>> tiny
{
    'Dda': 2,
    'Dmax': 7,
    'Dt': [[0, 0, 0], [0, 2, 25], [0, 25, 2]],
    'Dtinf': 25,
    'Dv': [0, 2, 1],
    'Na': 2,
    'Ne': 5,
    'Nec': [0, 2, 2],
    'Ni': 2,
    'Np': [0, 80, 80],
    'Nv': 2,
    'Nvmax': 2,
    'Ov': [0, 1, 2],
    'Pr': [0, 100, 100],
    'Ta': [0, 9, 13],
    'Td': [0, 8, 12],
    'Ty': [1, 1, 0, 0, 0],
    'Va': [0, 1, 2],
```

```
    'Vh': [1, 1, 1, 1, 2]
}
```

Paper and code submission

This project consists of two parts. You may submit one report for both parts, but please write two Python scripts.

Part 1: Basic resolution

First, define the variables of the problem. You may associate a variable to each flight taken by a staff member on a given day.

- **Constraints definition** (50%)

  Write and explain the code for each constraint in the report.

  - Flights assigned to `0` (no flight) must be at the end of the schedule. `[1, 0, 2]` could be equivalent to `[1, 2, 0]` but only the second one is valid.
  - If a staff member flies on that day, then (s)he leaves home and returns home.
  - A staff member may chain two flights iff :
    * first flight's destination airport and second flight's origin airport cities match;
    * there is enough transfer time between flights.
  - A staff member may not exceed his/her total time out of home;
  - Each flights requires at least two pilots and enough cabin crew members.

- **Problem optimisation** (5%)

  Start solving the constraint satisfaction problem before trying to optimise anything. You will probably go through a lot of trial and error before things work properly.

- **Answer the problem** (10%)

  To maximise your grade, please post-process and pretty-print the results of the optimisation and check they are consistent with the constraints. **If you find inconsistencies you cannot solve, please point them**.

  For each staff member, list the flights (s)he will be flying with cities and take-off/landing times.

  Valid example:

```
Staff number 0: 1 [1, 2] 2 [2, 1] scheduled at [8, 9] [12, 13]
Staff number 1: ...
```

For each flight, list the pilots and cabin crew members flying (together with the required staffing of the aircraft).

Valid example:

```
Flight number 1: pilots {0, 1} cabin crew {2, 3} (≥ 2);
Flight number 2: ...
```

Part 2: A more efficient modelling with CSP and MILP

Since bigger instances will require too much computing time, let's try something smarter:

1. First use a CSP solver to compute all possible paths between cities;
2. Then compare performances between a CSP and a MILP solver as we no longer assign staff members but groups of staff members to each path.

Warning.

Three instances are given (tiny, small, normal). Start debugging with the small instance; when it works, try the normal instance: you will be graded based on the results of the **normal instance**.

- **Preliminary questions** (10%)
  1. Why would it help to solve the problem for groups of staff member rather than individuals. Which mathematical principle enters into account?
  2. Try to explain why CSP better suits (compared to MILP) the path computing part of the problem.
- **Compute all possible paths** (10%)
- **CSP and MILP resolution** (15%)
  - Assign a group of staff members to each path;
  - Compare the computing time with CSP and MILP resolutions (use the `PuLP` library for MILP);
  - Post-process and pretty-print the results. Comment.