Constraint Programming Tutorial 2: Set Partitioning Problem

The SPP (*Set Partitioning Problem*) is an optimization problem which can be formulated with boolean variables $x_j \in [0, 1]$:

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{suibject to} \quad \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall i \in [1, m] \text{ and with } a_{ij} \in [0, 1] \tag{2}$$

The SPP can be used to model the aircraft rotation scheduling problem that must be solved by airlines. Each row ($\forall i \in [1, m]$) corresponds to a flight to be scheduled, and each column ($\forall j \in [1, n]$) represents a possible rotation for a crew. The (a_{ij}) constant matrix is defined by $a_{ij} = 1$ if flight i is carried out by rotation j, and by $a_{ij} = 0$ otherwise. The objective is to find a subset of rotations that **covers all the flight once** with **minimal** total cost, the cost of each rotation j being noted c_j.

Data for two instances, `toyrotations.ml` and `rotations.dat`, can be found on the e-campus server.

1. Use FaCiLe to implement a solver of the following instance (described with OCaml values in `toyrotations.ml`):

Rotation:	1	2	3	4	5	6	7	8	9	10
Cost:	5	1	2	1	4	1	3	2	1	2
flight 1	1	0	1	0	1	0	0	0	0	1
flight 2	1	1	1	1	1	1	0	1	0	0
flight 3	1	0	0	1	0	1	1	1	1	0
flight 4	1	1	0	0	0	0	1	0	1	1

   Indications:

   - *The main function (e.g.* `solve`*) will take two parameters: the cost vector and the rotation matrix.*
   - *Dynamically print the number of backtracks with the* `control` *optional parameter of function* `Goals.solve`*.*
   - *For each solution found, print its cost and the execution time (parameter* `solution` *of function* `Goals.minimize`*).*
   - *Also print the execution time at the end of the search (proof of optimality).*

2. The file `rotations.dat` contains the data of a real instance of larger size. The data are described according to the following format:

   - number of flights (m) and number of possible rotations (n) on the first line of the file;
   - then each line corresponds to a rotation (column):
     - its cost first;
     - then the number of 1s in the column (i.e. the number of flights covered by the rotation);

  - then a number to ignore (the "name" of the column);
  - and then the number of each row/flight (*indexed from 1 to m*) fixed to 1 for this column.

The best solution for this instance has cost 11307.

Indications:

- `open_in filename` *returns an input channel to read file* `filename`*.*
- `Scanf.fscanf ch format f` *reads the formatted data specified by* `format` *and returns the application of function* `f` *to the resulting arguments.*

For example, to read two integers with an arbitrary amount of spaces, tabulations or new lines (before, inbetween or after the numbers) at the beginning of file `filename`*:*

```
let ch = open_in filename in
let (m, n) = Scanf.fscanf ch " %d %d " (fun m n -> (m, n)) in
...
```

3. To improve the efficiency of the search, modify the variable instantiation strategy by first trying value 1 before 0. Justify this choice.