Constraint Programming Exam

Nicolas Barnier

Duration: 2h – All documents allowed – Access to the Web is forbidden

No anonymous numeric constants allowed in the source code!!!
All quantities must be named (except 0 and 1 of course).

The **commented** source code corresponding to your answers must be uploaded **before 12:15**. Follow the submission link in the exam section of the course page on e-campus.

This subject has 2 pages.

Seating Arrangement

To organize the seating arrangement at a wedding with n guests and m tables, several requirements must be taken into account:

- some pairs of guests must be at the same table (because they are couples for example);
- others must not be at the same table (because they hate each other for example);
- female-male parity must be respected as much as possible at each table.

We suppose that the tables all have the same number of seats k with $m \times k = n$ (i.e. $n \mod m = 0$).

An instance (`wedding.txt`) of this problem can be found on e-campus where are specified:

- the number of guests n and the number of tables m on the first line;
- the number of compatibility constraints on the second line;
- then there is one line for each of these constraints where guests are represented by a number in $[0, n-1]$ and the type of the constraint, compatibility or incompatibility, noted respectively by 1 or 0. For example:
  `1 3 0`
  means that guests 1 and 3 are incompatible (must not seat at the same table), whereas:
  `5 18 1`
  means that guests 5 and 18 must be seated at the same table.
- guest's gender (female or male) is represented on the last line by a 1 for female and 0 for male. For example:
  `1 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0`
  means that the first guest is a female, the second one a male etc.

Solve this problem with FaCiLe by answering the following questions:

1. **[2pt]** Define the boolean (`0/1`) decision variables matrix `table` where `table.(i).(j)` will be instantiated to 1 if guest j is seated at table i and to 0 otherwise.
2. **[4pt]** Add the main structural constraints of the problem:
   - the number of guests at each table is k;
   - a guest is seated at exactly one table.

3. **[3pt]** Define auxiliary variables `guest` where `guest.(j)` represents the table at which guest `j` is seated, according to the following expression defined for any j:

$$\sum_{i=0}^{m-1} i \times table_{i,j}$$

4. **[2pt]** Add the compatibility constraints using the auxiliary variables previously defined.

5. **[3pt]** Add a labeling goal that assigns the boolean variables `table` to find a solution to this problem with a carefully chosen *value ordering* to perform an efficient search. Justify your choice.

6. **[2pt]** Execute the search to provide a solution and print the number of backtracks.

7. **[4pt]** Males and females numbers must be balanced as much as possible at each table. Define auxiliary variables to represent the discrepancy (i.e. the absolute value of the difference) between the number of males and the number of females at each table. Then define the cost as the maximum difference for all tables and optimize the solution to the problem.
   Remark: Function `Arith.abs: Arith.t -> Arith.t` *can be used to return the absolute value of an arithmetic expression.*

```
barnier@venar:~$ ./wedding.opt wedding.txt
2 bt
cost=3 in 0.00158596s:
table 0:  0  1  2  5 18
table 1:  3  4  6  7  8
table 2:  9 10 11 12 13
table 3: 14 15 16 17 19
12 bt
cost=1 in 0.00319099s:
table 0:  0  1  2  5 18
table 1:  3  4  6  7  8
table 2:  9 10 12 13 14
table 3: 11 15 16 17 19
4570 bt
Proof in 0.0838151s
```