CSE 307 - Relational Programming

TP4 Disjunctive Scheduling in CLP(FD)

François Fages (Francois.Fages@inria.fr)

Bachelor of Science - Ecole polytechnique

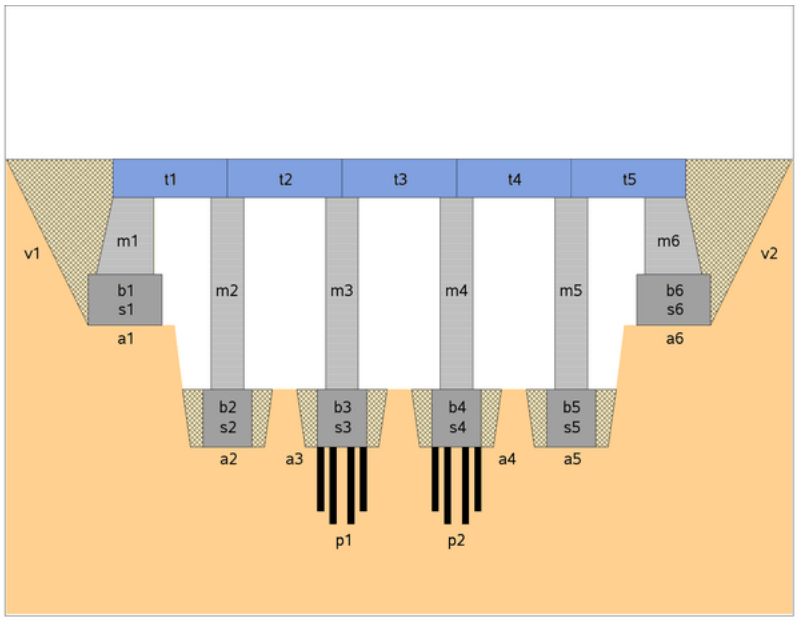
1. Building bridges for the metro of Saclay plateau

Let us consider the problem of scheduling a set of tasks (called jobs) for constructing a bridge in order to **minimize the delivery date**.

The tasks for the construction of the foundations, pillars and beams of the bridge must satisfy some obvious **precedence constraints**, using **non-strict inequality constraints** between the end date and the starting date of another task.

In addition, there are some extra **delay constraints** for the starting of the next task (e.g. time window constraints to let the concrete dry but not too much before building the next level).

Furthermore, some tasks have to use the same resources (excavator, pump, carterpillar, etc.) which creates **mutual exclusion constraints** and makes the general disjunctive scheduling problem NP-hard.



The file `bridgeData.pl` contains

- The list of task names
- The list of their durations
- The list of precedence constraints between tasks
- The list of tasks using each resource
- The list of either maximum or minimum time separation constraints between either start or end dates of two tasks

To solve this disjunctive scheduling problem, you will use the CLP(FD) library of SWI-Prolog which provides constraints on integers for expressing

- domains of variables `V in 0..3`
- precedence constraints `Sx + Dx #=< Sy`
- mutual exclusion constraints (`Sx + Dx #=< Sy) #\/ (Sy + Dy #=< Sx)`
- reified constraints `B #<==> (Sx + Dx #=< Sy)` the negation of a boolean is `#\ B`
- sum constraints `sum(+Vars, +Rel, ?Expr)`
- etc.

```
?- use_module(library(clpfd)).
true.

?- [A,B,C] ins 0..3, sum([A,B,C], #=, X).
A in 0..3,
A+B+C#=X,
X in 0..9,
B in 0..3,
C in 0..3.
```

You will create a file `bridge.pl` which loads the file `bridgeData.pl` and uses its predicates to define new predicates to solve the problem by answering the following questions.

Question 1. Define a predicate `starting_dates([S1,…,Sn])` to create the list of decision variables giving the starting dates of the tasks, with domain the interval 0 to the sum of the durations of all tasks.

Question 2. Define a predicate `job(([S1,…,Sn], Name, Start, Duration)` to ease access to data by giving the starting date variable and the duration of a job given by its name.

Question 3. Define a predicate `post_precedences([S1,…,Sn])` to post the precedence constraints on the starting date variables of the tasks.

Question 4. Define a predicate `post_delays([S1,…,Sn])` to post the delay constraints beteen the tasks.

Question 5. Define a predicate `schedule_deterministic(L)` to minimize the end of the project (starting date of end task `pe`). **The minimum end date should be 60**.

Question 6. Using disjunctions `#\/` define a predicate `post_disjunctions([S1,…,Sn])` to post mutual exclusion constraints between the tasks using the same machine.

Question 7. Define a predicate `schedule(L)` to **find one solution**. Remark that minimizing the end date might take too long in this approach based on enumerating the starting dates.

Question 8. Define the predicate `post_reified_disjunctions([S1,…,Sn],[B1,…,Bm])` to post the mutual exclusion constraints with reified constraints using one Boolean variable for each mutual exclusion constraints.

Question 9. Define a predicate `schedule_reified(L)` to solve the optimization problem for minimizing the end of the project `pe` by labeling the Boolean variables first, then the starting dates. **The optimal end date should be 104**.

Question 10. Explain the existence of several optimal solutions.