University Course Timetabling using CHR

Scheduling a set of courses within a given number of rooms and time periods.

Difficult and time-consuming Task

How to create a timetable by hand:

- Generation of a timetable of a term on base of the previous year's timetable considering:
  - Information on new courses
  - Wishes of teachers
  - General constraints
- Goal: *changing as little as possible*
  - Reduces the amount of work
  - Ensures acceptance

Requirements

- Must be satisfied:
  - Courses of a teacher do not clash
  - Courses of a unit do not clash
  - Courses of undergraduate studies do not clash
  - Do not schedule professors' courses for Monday afternoon
- Should be satisfiable:
  - Courses of graduate studies do not clash
  - At least a one hour break between two courses of a teacher
  - Teacher's wishes
  - A day break between the lectures of an offering
  - All courses should be scheduled between 9am and 6pm.

Modeling the Problem as PCSP

- One Variable for each course (Starting time point)
- Variable's domain consists of the whole week: e.g. 9 denotes 9am on Monday
- Constraints:
  - *No-clash constraints* demand that a course must not clash with another one
  - *Time constraints*: e.g. to express teachers' preferences
  - *Spreading constraints*: e.g. at least one day between a course and another one
- Hard and Soft Constraints
- A solution
  - satisfies all hard constraints
  - minimizes the total weight of the violated soft constraints

How to Handle Soft and Hard Constraints?

- Domain of variables: list of value-assessment pairs.

  `X::[(3, 0), (4, 1), (5, -1)]`: `4` is encouraged

- Propagation of a soft constraint: Changing assessment
  - Time constraint with weight `2` stating that `3` should be assigned to `X`
  - Increase the assessment for period `3` in the domain of `X`: the new domain of `X` is `[(3, 2), (4, 1), (5, -1)]`.
- Propagation of a hard constraint: Remove values from the variable's domain.
- The Search strategy uses the assessments to choose values.

Basic Constraints

- `domain(X, D)`: `X` is associated to the domain `D`.
  `D` is a list of pairs (Value, Assessment).
- `notin(X, L, W)`: Variable `X` must/should not occurs in `L`.
- `in(X, L, W)`: Variable `X` must/should occurs in `L`.

Constraints for Timetabling

- `start(c, S)`: the course `c` starts at the period `S`.
- `domain(S, D)`: Course associated with `S` take place at a starting time occuring in `D`: .
- `notin(S, L, W)`:
  - `W = hard`: the course associated with `S` must not start at any period appearing in `L`.
  - `W` is a number: the assessment for the periods appearing in `L` and not yet removed from the domain of `S` should be decreased by `W`
- `in(S, L, W)`: analougous
- Application-level constraints can be translated into `in` and `notin` constraints.

The CHR Solver

- Propagation of `notin` constraints:

```
domain(S, D), notin(S, L, W) <=>
  (
     W = hard
  -> domain_subtraction(D, L, D1),
  ;  decrease_assessment(W, L, D, D1)
  ),
  domain(S, D1).
```

- Propagation of `in` constraints:

```
domain(S, D), in(S, L, W) <=>
  (
     W = hard
  -> domain_intersection(D, L, D1),
  ;  increase_assessment(W, L, D, D1)
  ),
  domain(S, D1).
```

- A domain is reduced to the empty list:

```
domain(_, []) <=> fail.
```

Translation of `no_clash` Constraints

`no_clash(W, Cs)`: The courses in the list `Cs` must or should not clash depending on the weight:

```
no_clash(W, Cs) <=>
  Cs \= [_],
  select_ground_var(Cs, X, CsRest)
  |
  post_notin_constraints(W, X, CsRest),
  no_clash(W, CsRest).
```

- `select_ground_var(Cs, X, CsRest)` selects a ground variable `X` from `Cs`.
- `post_notin_constraints(W,X,CsRest)` produces `notin` constraints, one for each course in `CsRest`.

Plan Generation

- Each course will be associated with a `domain` constraint: Assessment is `0` for every value.
- All requirements will be translated into `in`- and `notin` constraints
- Search Procedure:
  - Variable selection: *first-fail* (Variable with the smallest domain)
  - Value Selection: *best-fit* (Value with the best assessment)