Mestrado (Integrado) em Engenharia Informática
Universidade Nova de Lisboa

Knowledge Representation and Reasoning
Second Project (due on December 7)

– Test/Exam Scheduling –

The tests/exams for the different courses of a department need to be scheduled taking into consideration information on the individual courses, including the enrolled students, the relevant time frame, and the indications from the responsible professors on when the tests may take place.

In more detail, university courses are composed of several academic years, each of which has two terms, and to each term and year specific courses are associated. To avoid that students have several tests/exams on the same day, and given that some students take tests/exams from different academic years, these tests/exams from different academic years need to be scheduled simultaneously.

The tests/exams are scheduled within given periods of time that can never overlap, each consisting of a number of days, and here we assume that each course has two tests and one exam, which are to be realized only in the according period of time. In addition, for each of the two tests, the responsible professors always indicate possible periods of dates for the tests, but only within the according periods of time of the respective test. For the exam, no such indications are admitted.

Objective The objective is to provide a schedule for a number of courses, taking into account the restrictions for them, that no student should do more than one test/exam per day, and that tests/exams from the same academic year with joint enrolled students should not occur on consecutive days. For larger numbers of courses and restricted time frames, this requirement may need to be relaxed, aiming for assignments that have as few students as possible with tests from the same year on consecutive days.

Data Factual information in this domain includes facts over `courseYear/2` that represent which course is given in which year of the degree; facts over `enrolled/2` that indicate the number of enrolled students per course, facts over `overlap/3` that indicate for two courses how many enrolled students they share, facts over `days/2` that provide which days are admissible for each of the test/exam periods, and facts over `pref/4` that indicate for a course and test the desired possible earliest and latest date. Example files are provided with testDates.txt for admissible dates, with prefs.txt for possible dates, and different NAMEData.txt files that provide info on the course data, with NAME varying among 1. lei (3 year undergrade course), 2. mei (2 year master), 3. miei (5 year integrated master), 4. bachelor (combining 1. and 3. on data from the first three years), 5. master (combining 2. and 3.), and 6. all combined.[1]

A correct solution in file scheduling1.lp should output something similar to the following:

```
clingo scheduling1.lp leiData.txt testDates.txt prefs.txt
clingo version 5.7.1
Reading from scheduling1.lp ...
Solving...
...
Answer:  4
assign(alga,2,1207) assign(am1,e,107) assign(alga,e,116) assign(ip,1,1031) assign(ip,2,1210)
assign(ip,e,114) assign(sl,2,1214) assign(sl,e,109) assign(am1,2,1217) assign(finf,2,1216)
assign(aed,2,1218) assign(aed,e,106) assign(finf,e,121) assign(fso,e,104) assign(lc,e,111)
assign(finf,1,1028) assign(lc,2,1203) assign(fso,2,1205) assign(cgi,e,107) assign(es,e,109)
assign(ia,e,118) assign(rc,1,1026) assign(rc,2,1202) assign(cgi,2,1209) assign(rc,e,113)
assign(ia,2,1214) assign(es,2,1219) assign(alga,1,1105) assign(lc,1,1108) assign(cgi,1,1109)
assign(am1,1,1021) assign(es,1,1023) assign(fso,1,1024) assign(sl,1,1025) assign(aed,1,1016)
assign(ia,1,1014) Optimization:  0
OPTIMUM FOUND
```

Note that dates are represented simply as number MonthDay, with January being 1, a single digit.

[1] While the course data is from this winter term, the preferences are generated and do not correspond to reality.

Group 1

Develop a solution based on Answer Set Programming that, for any given number of courses with their data on enrolled students and overlapping enrolements, dates, and individual time restrictions for the courses, returns a schedule for all test/exam periods that admits no students with more than one test on the same day, and that aims to avoid students having to take tests from the same academic year on subsequent days.

Group 2

Extend your previous solution based on Answer Set Programming in such a way that also the rooms are allocated for the tests and exams. For simplicity, we assume that there is only one timeslot per day, i.e., all tests of a same day are done at the same time. For this, you may use information about available rooms (in room.txt) that presents a number of rooms with their capacity (you can assume that for the given examples this suffices). The objective here is to assign only the minimally necessary number of rooms per test/exam.

Optional Optionally, for Group 1, you may in addition also minimize the number of students that have tests/exams on subsequent days from subsequent academic years (for undergrad degrees with less diversity of course, this is possible).

Also, you may further optimize the usage of rooms in Group 2 by ensuring a minimal number of used rooms, but at the same time, avoid too much empty space in the allocated rooms.

In addition, you may introduce further additions/variations in either group to improve your project, but ensure that any such addition (by you or from the suggested ones above) is done in a separate file.

Hints

- Note that there is an archive `scheduling.zip` available for download from CLIP containing the indicated instance data you can use for testing.
- It is recommended that you create some example files yourself to test particular situations.
- For both groups, the larger the instance, the more important it is, that your implementation is efficient. This may have a huge impact on performance.
- A decent solution after a reasonable amount of time is better than waiting forever for clingo to terminate (for your project as well as in practice).
- Independent problems may be solved separately. The archive also contains a python utility file (and an example test file) you may optionally use to that end.

Deliverables

You should send a single zip file labelled with your student numbers (in groups of two students) to `mkn@fct.unl.pt`. The file should contain:

- One file for each ASP program developed, labelled appropriately (e.g. scheduling1.lp, etc.).
- Any test files you may have used, labelled appropriately.
- Any relevant additional material you may have created.
- A report (single PDF file) which should include the commented code of all ASP programs used (with references to the corresponding files), explaining what you did in particular in Group 2, and the encoding optimizations applied, as well as any supplementary material.

Finally, note that different groups have to submit different solutions; in case of plagiarism all groups involved will fail the project.