TimTab

The problem is to time table lectures for the School of Computing Science (SoCS). Generally, timetabling is three dimensional: decide who does what (teaching allocation), decide where this is done (room booking), decide when things take place (scheduling). We are interested only in the scheduling, i.e. deciding when lectures take place.

There are obvious constraints, such as "a lecturer can only be in one place at one time". But there are many others, peculiar to SoCS and the University. Some of these constraints are described in the student handbook.

To describe our system (very much work in progress) I will describe first the input to the system. Then I will give a brief description of how the system models the timetabling problem as a Constraint Program (CP)

The Input Files

Our timetabling system takes as input the following files

courses.txt

This file contains the courses that are taught. It also gives a motif describing the allowed pattern of lectures. For example we might have an entry "ALG4 1 1 1" meaning that the course Algorithmics4 has 3 lectures in a week, each on a different day. Another example "ML 2 1" means that Machine Learning has 2 lectures on one day and then a third lecture on a different day. These motifs are in a state of flux (no standard or convention as yet) but are defined explicitly in the program that reads this file (and this should be documented and made more human-readable).

years.txt

This is maybe a tad controversial, but this is a file that gives the courses that take place in a year, i.e. the course that make up a year. Consequently the lectures that make up the courses in a year must be all different. For example "L2 ADS2 AF CS2 JOOSE1 JOOSE2 WAD" gives the courses in level 2 and "L1a 1CT 1P2 1Q1 1Q2" the courses that make up one track of level 1 and "L1b 1P1 1P2 1Q1 1Q2" the other track of level 1. Arguably, this file in a sense presents "specialisms" and might be combined into the file specialisms.txt (see below)

fixed.txt

Some lectures are already scheduled into time slots, examples being the level 1 lectures and most all of level 2. This file contains those lectures that have already been scheduled. What is of interest here is how we represent time (and this is used throughout). For example "ADS2 2TU11 2TH11 2FR13" means that ADS2 is taught in semester 2 on Tuesday at 11, semester 2 Thursday at 11 and semester 2 Friday at 1 o'clock. Another example is AF (Algorithmic Foundations "AF 1WE11 1FR11 1FR13" meaning it is taught semester 1 on Wednesday at 11, semester 1 Friday at 11 and semester 1 Friday at 1 o'clock. NOTE: the names of the courses in this file must already exist in courses.txt and the pattern/motif in courses.txt must be compatible with the allocated times in fixed.txt, although at present there is little verification to confirm this.

forbidden.txt
This file is, in some sense, the opposite of fixed.txt in that it gives times when lectures for a course cannot take place. For example "DAS 1MO16 1WE14 1WE15 1WE16" means that all of the lectures in the course DAS cannot take place at the times semester 1 Monday at 4 o'clock, Wednesday at 2 o'clock, ...

lecturerclashes.txt
This gives courses, given by the same lecturer, and therefore cannot occur at the same time. For example the entry "ADS2 CP" means that ADS2 and CP are taught by the same lecturer and none of the lectures that make up those courses can occur at the same time.

specialisms.txt
This gives sets of courses that make up a specialism (such as software engineering), consequently all of the lectures that make up a specialism must occur at different times (otherwise a student could not complete that specialism). This file is not yet fully defined and needs updating with respect to the student/course handbook.

prereq.txt
This gives the prerequisites for a course. For example "CSC2 CSC1" states that the prerequisite for CSC1 is CSC2. Note, it is assumed that this is only for courses in the same year, and this will force the courses into different semesters (in our example CSC1 into semester 1 and CSC2 into semester 2). It does not state prerequisites between years, as there is no need to do so (Consider course X in level 3 and course Y in level 4, and X is a prerequisite for Y. This is no reason to prevent X and Y occurring in the same semester).

The Model
The model is implemented using Choco2 in Java. The main classes of "things" is as follows

Lecture.java
This is the most basic object and is of course a lecture. A lecture belongs to a course. The Lecture contains a constraint integer variable, it's start time, and this is an integer in the range 0 to 79. The value 0 is 09:00, Monday, semester 1 and 39 is semester 1, Friday at 4 o'clock. Values 40 to 79 are then in smester 2. This class also defines potential constraints between lectures, such as sameDay, differentDay, sameSemester, nextSemester, and of course some of James Allen's temporal relations such as meets (used to create a 2 or 3 hour lecture), before, notEqual etc.

Course.java
This is a course and contains a list of lectures. This also contains the "motif" for that course, defining the pattern of the lectures in that course. There are also constraints defined for courses, such as noConflicts (meaning that no lectures in two given courses can occur at the same time), prereq (forcing lectures into different semesters). A course also has a smester, and this is a constrained integer variable with a 0/1 domain. The semester is constrained with the start times of lectures in that course. Also, a course belongs to a year.

Year.java
A year is a list of courses that make up a year. The year contains a two dimensional array of 0/1 variables called conflict, where conflict[i][j] = 1 iff the ith and jth course have some lectures that occur at the same time. Therefore we can force no conflicts between pairs of courses by setting array values, or we might allow conflicts and attempt to minimise these. In a sense, a Year is, as above, a specialism and might be better thought as such. There is also a one dimensional array called semester (maybe not a good name) and has one entry for each course in that year. Therefore semester[i] = 1 iff course[i] is in semester 2, 0 otherwise. Consequently, if we have n courses in a particular year we would like the semester array for that year to add up to n/2, giving a balanced curriculum. This is then used as a constraint or as an optimisation criteria.

TimTab.java
This is the timetable object where the schedule is produced, all files are read in, the model built and timetables outputted.

Reflections & Conclusion

There is still some fuzzy thinking regarding specialisation and years. I suspect that specialisations are important as they give us a set of course that must not conflict and should we well balanced across two semesters. However, a year might be all courses in a year (whatever that really is) and what we want is to minimise conflicts, thus maximising the choice for optional courses, and possibly also maximise balance across semesters.

We need to get all off the specialisms into our system. This requires an analysis of the SoCS handbook (or whatever it is called). We also need to make our naming of courses consistent, and friendly. Our input files also need comments, and this means modifying our file readers (but most of this has been done).

This is all very much work in progress. We have some problems with regard to years and specialisms. We also need to make, and verify input file consistency. Just now, the model attempts to find a satisfying schedule, when one might not exist. Therefore, we need to properly address optimisation with regard to minimising conflicts and maximising balance.

Patrick Prosser
01/07/2015