FULLY HEURISTIC TIMETABLING DRIVEN BY STUDENTS' FEEDBACK ON AVAILABILITY

Filipe Sousa, Albano Alves

[1]Instituto Politécnico de Bragança
Campus de Santa Apolónia, Apartado 1134, 5301-857 Bragança, Portugal, tel.: +351 273303036, E-mail: filipe@ipb.pt
[2]Instituto Politécnico de Bragança
Campus de Santa Apolónia, Apartado 1134, 5301-857 Bragança, Portugal, tel.: +351 273303001, E-mail: albano@ipb.pt

Abstract

The optimal schedule of lectures and exams is critical in higher education institutions. But it is a very time consuming task for those who are in charge of planning academic activities. For this reason, a considerable attention has been devoted to automated timetabling.

Because timetabling solutions are highly dependent on the way institutions are organized, it is common that each institution develops its own platform. At the Instituto Politécnico de Bragança we have evaluated several existent solutions but we have concluded that no particular solution fulfills the totality of our requirements. That leads us to the development of a new platform in order to handle all constraints we consider relevant and to incorporate all knowledge we have accumulated from manually scheduling lectures for many years.

In particular, we are interested in the optimal scheduling of lectures from the students point of view. Our institution enrolls many student workers and many students from outside our region and, in addition, our study plans are composed by a majority of compulsory courses, which makes difficult the scheduling of all lectures in a manner that the majority of students can attend the lectures they want or they have to. Therefore we decided to include students' availability as a soft constraint.

The platform we are still improving produces final timetables through an algorithm that implements some heuristic scheduling techniques and runs on an HPC environment. From time to time the system tries to relax students' constraints by asking students to rethink and reintroduce their availability within certain limitations.

Introduction

The timetabling problem consists in fixing a sequence of meetings between teachers and students in a prefixed period of time (typically a week), satisfying a set of constraints of various types.

The manual solution of the timetabling problem usually requires several days of work and the final solution may be unsatisfactory because it is a highly complex task to verify all constraints.

For the above reason, a considerable attention has been devoted to automated timetabling. A large number of variants of the timetabling problem have been proposed in the literature, which differ from each other based on the type of institution involved and the type of constraints. In [1] timetabling problems are classified in three main classes:

- school timetabling - the weekly scheduling for all the classes of a high school, avoiding teachers meeting two classes at the same time, and vice versa;
- course timetabling - the weekly scheduling for all the lectures of a set of university courses, minimizing the overlaps of course lectures having common students;
- examination timetabling - the scheduling for the exams of a set of university courses, avoiding to overlap exams of courses having common students and spreading the exams for the students as much as possible.

At the *Instituto Politécnico de Bragança* (IPB) we are developing a platform that integrates examination timetabling and a mix of school timetabling and course timetabling. Our interest in mixing school timetabling and course timetabling concepts is due to the fact that we consider grouping students in classes a better solution, but sometimes students will belong to several classes. Since our study plans include a higher number of compulsory courses, we can easily group students and create the context of a class, which is more pedagogically effective, but we also have to consider the individual choice of courses by students, mainly because students that don't pass final examination (at the end of a specific semester) have to repeat some courses a year later.

At IPB we developed a completely new timetabling software - the GAL platform - that allows us to: 1) specify all type of constraints using simple GUI interfaces, 2) manually schedule lectures, 3) verify if every constraint is satisfied. In addition, this platform supports the definition of new functionalities through plugins; since 2004 we have been developing new plugins to extend the automation capabilities of the platform in order to achieve a 100% automated solution for our timetabling problems. Our main focus has been on the implementation of heuristic scheduling techniques based on the knowledge we have accumulated from manually scheduling lectures for many years.

At present we are trying to enhance the overall functionality of the GAL platform by allowing students to specify their own availability, in particular because IPB enrolls many student workers and many students from outside our region. To guarantee that lectures are scheduled according to the students' preferences, we first ask each student to point out his availability, using a web interface, and then the system tries to schedule lectures and detects severe conflicts due to students' availability restrictions. Subsequently the platform tries to find a

compromise solution by presenting students, using email messages, some alternatives for the schedule of lectures and by asking them to reintroduce their availability. The process is repeated until an acceptable solution is obtained and it has some similarities with the one already suggested in [2].

Timetabling constraints

An important phase in the process of scheduling lectures or exams is the correct definition of the problem. In the GAL platform the following data may be introduced through GUI interfaces:

- hour by hour availability for teachers, students and classrooms;
- specification of all resources available at each classroom and required by each course;
- students' registration for courses (crucial for detecting the overlapping of lectures and for choosing a correct size classroom);
- definition of course lectures (including the number and duration of time blocks and the responsible teacher);
- minimum and maximum interval between two consecutive lectures of the course.

For what concerns the introduction of students' availability, the platform offers a web interface, as presented in figure 1, where each student is able to indicate what periods he will not be available during the week for attending lectures.

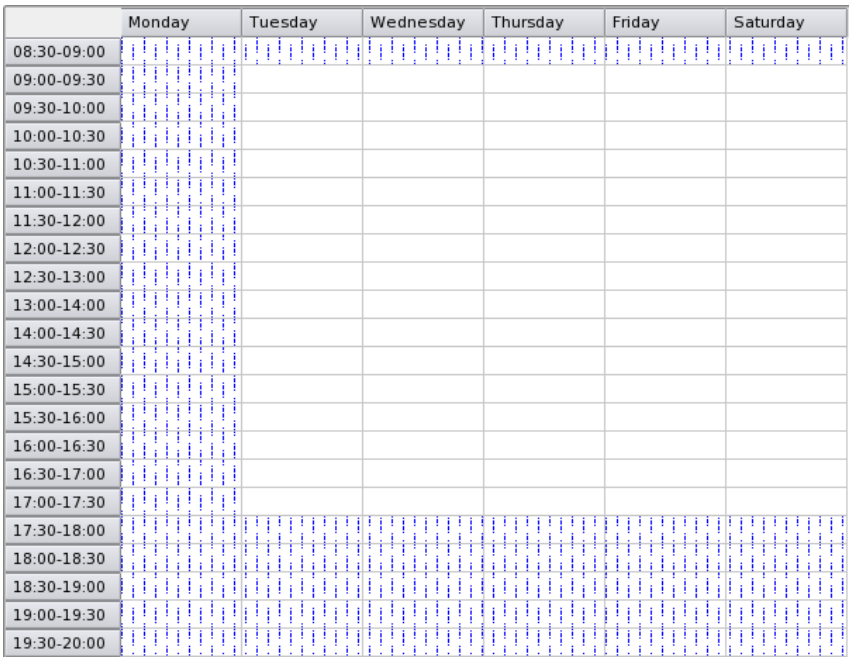

Figure 1 - web interface for the introduction of availability constraints.

Initial solution

The GAL platform uses a simple heuristic algorithm to produce a feasible assignment of lectures. The algorithm, which uses some strategies presented in [3], mimics the manual process we had used before the introduction of automation capabilities in the GAL platform and may be summarized as follows:

- lectures are sorted according to the expected difficulties in the scheduling process; as in the manual solution, the system tries to first schedule the more constrained/conflicting lectures;
- for each lecture, all possible slots for its allocation are calculated, without considering classroom requirements;
- for each possible slot, for a specific lecture, all available and adequate classrooms are identified; if classrooms are available, the one that permits to minimize the waste of resources is selected and the lecture is scheduled.

The algorithm also includes many “tricks” that we have learned from manual timetabling, e.g. selecting time slots near lunch time to produce more compact timetables and to increase the availability of classrooms. However, it is almost impossible to obtain a complete schedule, even if students' availability constraints are ignored.

The system needs to rerun several times the base algorithm in order to find a feasible solution. For each iteration, the system removes some scheduled lectures from the timetable and tries to schedule other lectures that could not be assigned at earlier attempts.

To achieve good performance, we have parallelized the algorithm by using distributed data structures implemented directly above the MX library [4], a low-level message-passing system for Myrinet Networks that allows to efficiently exploit 10-Gigabit network interface cards.

Solution refinement

The original availability constraints specified by students constitute a severe limitation to the scheduling process. In fact, students tend to point out large amounts of unavailable slots without any determinism. Therefore the join of all unavailable slots specified by the students of a class may disallow the scheduling of any lecture.

To find an initial feasible solution, the GAL platform ignores students' availability constraints that are not shared by the majority of the students of the class. That way lectures are assigned against students' preferences and a negotiation stage must be started.

During the negotiation stage, the system sends email alerts to all students whose availability was not respected. Those students can then specify new availability constraints, considering the guidelines presented by the system; the GAL platform "encourages" students to relax availability constraints.

At the end of the negotiation stage, the system starts a refinement stage, comprising the following steps:

- the new constraints are examined to verify if the existing solution is now satisfactory;
- if the solution is not satisfactory, all conflicting lectures are removed and the main scheduling algorithm is repeated.

When the system finds that the availability constraints of teachers and classrooms represent a significant limitation, an alert is sent to the timetabling team; if necessary, availability constraints of teachers and classrooms may be relaxed by the administrator to benefit students.

The whole process is repeated until a satisfactory solution is obtained.

Discussion

Many authors believe that the timetabling problem cannot be completely automated. The reason is twofold: on one side, there are reasons that make one timetable better than another one that cannot easily be expressed in an algorithm. On the other side, since the search space is usually huge, a human intervention may bias the search toward promising directions that the system by itself may be not able to find.

The intervention of students in our platform, by interactively specifying availability constraints, brings two main advantages:

- students are faced with the real timetabling problem and they try to slightly adequate their availability to help the system to find a global solution;
- by keeping specific time slots as unavailable, a class forces the system to take another search direction and to produce a better timetable.

A self-evident extension to the platform would be to incorporate additional scheduling techniques, in addition to the fully heuristic strategy we had implemented. Genetic algorithms, for instance, could be an interesting approach to solve conflicting restrictions due to the students' availability constraints.

References:

[1] Andrea Schaerf, "A survey of automated timetabling", Artificial Intelligence Review, 13(2):87-127, Kluwer Academic Publishers, 1999.

[2] Tomás Müller, Roman Barták, "Interactive Timetabling: Concepts, Techniques and Practical Results", PATAT 2002 - 4th international conference on the Practice And Theory of Automated Timetabling, 58-72, 2002.

[3] P. A. Kostuch, "Timetabling Competition - SA-based Heuristic", 2003.

[4] Myricom, "Myrinet Express (MX): A High-Performance, Low-Level, Message-Passing Interface for Myrinet", Myricom, Inc., 2006.