10th International Conference of the Practice and Theory of Automated Timetabling
PATAT 2014, 26-29 August 2014, York, United Kingdom

Integrated Student Sectioning

Jeffrey H. Kingston

Abstract The assignment of students to sections has always seemed different from the rest of timetabling. Courses demand times, teachers, and rooms, but not students; instead, students demand courses. This paper shows how to remove this difference and integrate student sectioning with the rest.

Keywords Timetabling · Student sectioning

1 Introduction

When demand for some course is greater than the maximum class size, the course has to be broken into *sections*: copies of the course. Each student who enrols in the course must be assigned to a section, and this is the *student sectioning problem*. It arises both in universities [1,7] and high schools [2,5].

Student sectioning has always seemed different from the rest of timetabling. Courses demand times and resources (teachers, rooms, etc.), but not students; instead, students demand courses. This point is clarified below (Sect. 2).

One negative effect of this perceived difference is that good ideas developed on one side of the barrier may seem to be inapplicable on the other. For example, the author works with a data structure he calls the *global tixel matching* [4]. It keeps track of whether an instance's resources are sufficient to satisfy the demand for resources, both overall and at specific times. is useful when assigning teachers and rooms in high school timetabling [4], so it might be useful for students too—but not if student assignment is genuinely different.

Another negative effect is that solvers assign students in one way, and teachers and rooms in another, encouraging these tasks to occur in separate phases of the solver. The literature is divided on whether to assign students to sections first [1], or to assign times first and then timetable each student.

J. Kingston
School of Information Technologies, The University of Sydney, Australia
E-mail: jeff@it.usyd.edu.au

This paper breaks down the barrier by transforming student sectioning into conventional timetabling. The author knows of no previous publication of this transformation, but it is not likely to surprise researchers familiar with student sectioning. This paper's novelty lies more in its uncovering of the *utility* of the transformation: it allows good ideas from conventional timetabling to be applied to student sectioning, and it permits the whole timetable to be built in an integrated way, bypassing the problems caused by separate phases.

2 The transformation

This section presents the transformation of student sectioning into ordinary timetabling via an example. It will be clear that it works in general. A few simple extensions are also presented. The transformation would be carried out by software when preparing for solving, not by the end user.

A university has two Chemistry laboratories (rooms) for senior students. A laboratory event occupies four hours, and may start at 9am or 2pm each week day. This makes a total of 20 events per week (10 per week in each of the two laboratories). Each laboratory can hold up to 20 students, giving capacity for 400 students. In fact, the university has decided to have only 17 events, the minimum required to accommodate the 325 currently enrolled students.

A conventional representation consists entirely of 17 events of the form

$$C_j = \langle 4\ T92, 1\ SenChemLab, 1\ SenChemTeacher, \leq 20\ SenChemStudent \rangle$$

The first item of the tuple is a demand for 4 consecutive hours beginning at one of the elements of set $T92$, defined elsewhere to be the 9am and 2pm times; the second is a demand for one laboratory from the set $SenChemLab$ of senior Chemistry laboratories (this set has two elements); and the third demands a senior Chemistry teacher in the same way. The solver is required to assign an appropriate starting time, lab, and teacher to this event. Various constraints must be obeyed: the resources must have no clashes, be employed only when available, and so on. These other constraints are not our focus here.

The fourth item allows up to 20 students from the set $SenChemStudent$ to attend. The '$\leq$' sign marks the difference from the other requests: it is a defect if any of those are not met, but 20 students is just the maximum. Furthermore, the same room or teacher may attend several of these events at different times; but each student should attend exactly one. Clearly, the student assignments are anomalous, and this is what motivates this paper.

The idea of the transformation is to create an event for each student s_i in $SenChemStudent$ in which s_i meets with a *seat* in a laboratory. A seat, quite concretely, is a piece of furniture capable of holding only one student at any one time. Each laboratory has 20 seats, making 40 seats altogether, and these are the elements of set $SenChemSeats$. The event for student s_i is

$$S_i = \langle 4\ T92, 1\ SenChemSeat, s_i \rangle$$

where s_i indicates a preassignment of s_i to this event. Preassignments are standard in timetabling. If times and seats are assigned to these 325 events, and no seat has a timetable clash, then no laboratory will be overfull.

The original events C_j are retained in modified form:

$$E_j = \langle 4\ T92, 1\ SenChemLab, 1\ SenChemTeacher \rangle$$

Assignments of students and teachers might occur in all 20 possible events, whereas the university has decided to have only 17. So three events have to be excluded, and this is done by adding three events of the form

$$R_k = \langle 4\ T92, 1\ SenChemLab, 20\ SenChemSeat \rangle$$

At the times occupied by these events, one fewer laboratory and 20 fewer seats will be available. There is nothing to force the selected seats to come from the selected laboratory, but if they don't, a simple reassignment after solving ends fixes that problem. (A specific laboratory and its 20 seats could be preassigned to one R_k, but it is not clear whether to preassign them to several R_k.)

Several extensions can be added. If students work in preassigned pairs, then each student event could request two seats and contain two students. The seats may not be adjacent in the same laboratory, but that is easily fixed during the seat reassignment after solving ends. At the author's university, senior Chemistry students attend two laboratory events each week. This can be handled by creating two S_i events per student.

There is a problem when there is no natural low limit to the number of events that may occur simultaneously, as is imposed by having only two laboratories in the example. Take a junior Mathematics course broken into 20 sections of 30 students each, held in ordinary rooms which are in plentiful supply. A pure transformation would require 600 seats, the great majority of which are unavailable. In practice, however, it would be safe (and perhaps desirable) to arbitrarily limit the number of events that occur simultaneously, coming close to the Chemistry example. It may be important to do this, to ensure that the transformed instance is not too much larger than the original.

Additional constraints, such as that each event have at least some number of students, or that students be distributed among the events as evenly as possible, must be imposed separately, in both the original instance and the transformed one. Representing them in the transformed instance is a potential problem, because the transformation fragments one event into many.

3 Solving

Conventional solvers first construct a solution and then repair it. There may be a first phase which assigns times, and a second which assigns resources. What would such a solver make of a transformed student sectioning instance?

If, after every event is assigned a time, the global tixel matching reports that sufficient resources are available at all times, then resource assignment for

the transformed events is trivial and must succeed (except for unpreassigned teachers, who may have constraints which make assignment difficult, with or without the transformation). So the student sectioning part of the problem is basically about constructing and repairing a time assignment which minimizes the 'insufficient resources' defects reported by the global tixel matching.

If it is considered expedient to group students into initial sections before solving [1], then this is *hierarchical timetable construction* [3]: placing the S_i into groups and constraining the elements of each group to be simultaneous.

Initial construction of a time assignment, with the global tixel matching as a guide, should find a timetable which makes a reasonable starting point for repair. Swapping a C_j with an R_k moves a section to a different time; moving or swapping an S_i moves a student to a different section. Both seem useful. There are also some natural larger neighbourhoods, the kind used by VLSN search [6]. An example is unassigning one student's events, then reassigning them using a tree search with intelligent backtracking. This has been done at the author's university for many years, but never published.

All this could go on at the same time as repairs are tried which improve other aspects of the timetable, producing an integrated solve rather than a series of separate phases. Doing it without the transformation is possible but much clumsier, since special arrangements would then have to be made to compute the equivalent of the current availability of seats at each time.

References

1. Michael W. Carter, A comprehensive course timetabling and student scheduling system at the University of Waterloo, Practice and Theory of Automated Timetabling III (Third International Conference, PATAT2000, Konstanz, Germany, August 2000, Selected Papers), Springer Lecture Notes in Computer Science 2079, 64–81 (2001)
2. Peter de Haan, Ronald Landman, Gerhard Post, and Henri Ruizenaar, A case study for timetabling in a Dutch secondary school, Practice and Theory of Automated Timetabling VI (Sixth International Conference, PATAT2006, Czech Republic, August 2006, Selected Papers), Springer Lecture Notes in Computer Science 3867, 267–279 (2007)
3. Jeffrey H. Kingston, Hierarchical timetable construction, Practice and Theory of Automated Timetabling VI (Sixth International Conference, PATAT2006, Brno, Czech Republic, August 2006, Selected Papers), Springer Lecture Notes in Computer Science 3867, 294–307 (2007)
4. Jeffrey H. Kingston, Resource assignment in high school timetabling, Annals of Operations Research, 194, 241–254 (2012)
5. Simon Kristiansen and Thomas R. Stidsen, Adaptive large neighborhood search for student sectioning at Danish high schools, PATAT 2012: Ninth international conference on the Practice and Theory of Automated Timetabling, Son, Norway (2012)
6. Carol Meyers and James B. Orlin, Very large-scale neighbourhood search techniques in timetabling problems, Practice and Theory of Automated Timetabling VI (Sixth International Conference, PATAT2006, Brno, Czech Republic, August 2006, Selected Papers), Springer Lecture Notes in Computer Science 3867, 24–39 (2007)
7. Keith Murray, Tomáš Müller, and Hana Rudová, Modeling and solution of a complex university course timetabling problem, Practice and Theory of Automated Timetabling VI (Sixth International Conference, PATAT2006, Brno, Czech Republic, August 2006, Selected Papers, Springer Lecture Notes in Computer Science 3867, 189–209 (2007)