Post-enrollment-based course timetabling with Moses: System demonstration

An IT system for academic resource planning at universities

János Höner · Gerald Lach · Erhard Zorn

Keywords Post-enrollment-based course timetabling · Automated timetabling · Integer programming · Resource planning · Higher education

1 Introduction

Since 2002, *innoCampus*, a department of the Technische Universität Berlin (TU Berlin), has been doing research on timetabling problems of various kinds. In the context of this work, an IT system called *Moses* has been implemented. Besides numerous administrative features, the system offers modules for examination, post-enrollment, and general curriculum-based university course timetabling problems. The post-enrollment-based course timetabling will be presented in this system demonstration.

In spring 2003, *Moses* was used for the first time at TU Berlin (more than 32,000 students) to distribute students to their tutorials (additional small exercise classes offered for courses with large numbers of students) in mathematics courses. Today, more than 80 large courses are using *Moses* to distribute their students into more than 1,000 tutorials.

János Höner
Technische Universität Berlin
Institute of Mathematics/innoCampus
E-mail: hoener@math.tu-berlin.de

Gerald Lach
Technische Universität Berlin
Institute of Mathematics/innoCampus and
MathPlan GmbH
E-mail: gerald.lach@mathplan.de

Erhard Zorn
Technische Universität Berlin
Institute of Mathematics/innoCampus
E-mail: erhard@math.tu-berlin.de

Due to its great success, *Moses* has been extended to solve the examination timetabling problem (for details see [2]) and the university course timetabling problem (for details see [3]). Since 2013, it has also been used at RWTH Aachen University (approximately 44,000 students) for the university course timetabling problem. Since 2015, *Moses* has been used at the Technical University of Munich (more than 39,000 students) for the examination timetabling problem. These universities are among the largest and most reputable universities in Germany. This paper focuses on the current deployment of *Moses* at TU Berlin.

2 Approach

Our approach to solving timetabling problems even for big universities (with more than 30,000 students) is based on the idea of combining powerful mathematical optimization algorithms with a clear workflow and easy-to-use graphical user interfaces that capture and support this workflow. Our experience shows clearly that for a successful optimization in the real world, this workflow is at least as important as a carefully designed optimization algorithm.

This is true especially in the case of the post-enrollment timetabling problem, where a large number of students are directly involved in the data input.

3 Process

The process of planning the small courses (here called tutorials) in the post-enrollment context involves two main user roles: On the one hand, there are lecturers or administrative staff responsible for the data concerning the tutorials (e. g. Which are the potential time slots for the tutorials? Which rooms are suitable?). On the other hand, there are the students who enroll for the tutorials and can state their preferred time slots. Besides these two roles, there is a planning supervisor who is in charge of checking the data and invoking the optimization after the data input phase end.

The whole process can be broken down into the following steps:

1. Preparation of the planning (supervisor)
2. Course data input (lecturers and administrative staff)
3. Student data input (students)
4. Data cleaning (supervisor)
5. Invoking timetable optimization (supervisor)
6. Publication of the timetable (supervisor)
7. Individual minor changes (lecturers and administrative staff)

What are the key properties of the system that facilitate this process significantly?

One key feature is the decentralized way of collecting the data. Each user involved in the planning process can input his or her own data whenever and

from wherever he or she wants, as long as it is in the corresponding data input phase.

To establish this key property three main requirements had to be fulfilled.

First, *Moses* is accessible from any modern browser on most devices.

Second, the data-input graphical user interfaces were designed to be self-explanatory and easy to use. They also provide a lot of feedback for the user to ensure a high quality of input data.

Third, a flexible and powerful user permission management plays a vital role in controlling the workflow outlined above. The organizational structure of most universities is more or less tree-like. This structure is mirrored by the system's permission management. The permissions module includes flexible control of who can grant which permissions to whom and the possibility of recombining permissions in an arbitrary manner.

Once the data has been collected and a timetable has been prepared successfully, the result is published and can be seen by all involved users. It is still possible to do minor changes by hand—e. g. moving a student from one tutorial to another or adding students who where unable to enroll in the tutorial during the data input phase.

The results of the optimized timetable can be used for further administrative tasks within *Moses*—e. g. it is possible to manage homework and test results for the students of one tutorial without having to create an additional list of participants. Furthermore, these homework results can be used to control the access to tutorials—e. g. students may have to successfully complete the tutorials of Lecture A to be admitted to the tutorials of Lecture B.

4 System Architecture

The system comes as a Java EE web application running on the latest version of the Glassfish Application Server. The user interfaces are based on JSF and the comprehensive JSF-based *Primefaces* library. Beyond that, modern responsive design (mainly based on the front-end framework *Twitter Bootstrap*) enables users to access the system from a device of their choice.

The optimization consumes a lot of processing power and, therefore, is carried out on a different machine than the one on which the application server is installed. A specialized distribution server communicates with the application server and controls the optimization processes. This distribution server has access to a certain number of optimization machines to which the scheduling tasks are assigned.

Typically, the IT landscape of a university—that is to say the diverse systems that form the university's overall IT system—is highly heterogeneous. That is why the *Moses* system offers various interfaces to other systems for importing and exporting data automatically, semi-automatically, or manually.

5 Optimization

The optimization model behind the system is a linear mixed-integer programming (MIP) model. The timetabling problem is converted into such a model and, thereafter, solved using a commercial MIP solver. Only our careful design of the model guarantees that even large timetabling problem instances are solved in an acceptable time span.

In contrast to many other planning solutions relying on heuristics, we are able to compute a truly optimal plan. Even if the computation time for obtaining an optimal plan exceeds our limits, we are able to tell exactly how far from the optimum we are in the worst case.

The mixed-integer programming model used for the optimization of the post-enrollment course timetabling problem is based on the model presented in [1]. Of course, some modifications had to be done to model the reality more accurately and we are working on a paper stating these modifications in detail.

5.1 Mixed-integer Program

The full problem is modeled as a mixed-integer program, where the objective function includes the respective priorities (for the different time slots) of the students and a number of penalty variables to capture the following cases.

- Tutorials that do not take place in their preferred rooms
- Student groups that are separated
- Tutorials that are too small (to assure almost equi-sized tutorials)
- Students that could not be assigned
- Tutorials that could not be scheduled

All remaining necessities are either handled in a preprocessing step by excluding corresponding variables from the optimization problem (e .g. no unsuitable rooms, no tutorials in time slots where they cannot take place) or modeled as constraints (e. g. not too many students in a room, not more than one tutorial in one room at a time).

5.2 Performance

The described model was used for the first time to plan the tutorials for the summer term of 2016 (April 2016). In this term, we planned 1,052 tutorials of 77 courses and assigned 8,417 students to 20,747 seats. The MIP solver was able to solve the problem to optimality in about 90 minutes.

5.3 Fallback Heuristic

To prevent *Moses* from failing even with much larger instances in the future, we are currently developing a heuristic as a fallback method. The main

idea of this heuristic is to draw a small sample of students and solve the full post-enrollment course timetabling problem only for this subset of students (the capacities of the rooms are adjusted according to the proportion of the sample). We then take the assignments of tutorials into rooms and time slots from this solution and distribute all students among these tutorials (student scheduling problem). The rationale behind this approach is the assumption that a randomly drawn sample of students captures the overall preferred time slots quite well and there is no need to optimize for the priorities of every single student.

6 Future Work

The bigger the system grew, the more obvious it became that modularization is necessary. A decomposition into pluggable modules is, therefore, the most urgent development task. Due to the big and heavily cross-linked database and the interconnected general structure, this is expected to be challenging work. The greatest advantage of a pluggable architecture would be that the system could be better adapted to the specific needs of the universities applying it. For the same reason, internationalization and pluggable branding are other goals worth pursuing.

References

1. Höner, J., Lach, G., Zorn, E.: An IP-based model for the post-enrollment-based course timetabling problem at TU Berlin. In: Z. Hanzálek, G. Kendell, B. McCollum, P. Šůcha (eds.) MISTA 2015. Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications, pp. 331–344. Prague, Czech Republic (2015)
2. Lach, G., Lach, M., Zorn, E.: Examination timetabling at Technische Universität Berlin. In: Z. Hanzálek, G. Kendell, B. McCollum, P. Šůcha (eds.) MISTA 2015. Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications, pp. 260–266. Prague, Czech Republic (2015)
3. Lach, G., Lach, M., Zorn, E.: Solving huge real-world timetabling instances. In: Z. Hanzálek, G. Kendell, B. McCollum, P. Šůcha (eds.) MISTA 2015. Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications, pp. 370–378. Prague, Czech Republic (2015)