University course timetabling with Moses: System demonstration

An IT system for academic resource planning at universities

Gerald Lach · Mirjana Lach · Julian Schick · Erhard Zorn

Keywords University course timetabling · Automated timetabling · Integer programming · Resource planning · Higher education

1 Introduction

Since 2002, *innoCampus*, a department of 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 latter 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

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

Mirjana Lach
Technische Universität Berlin
Institute of Mathematics/innoCampus
E-mail: mlach@math.tu-berlin.de

Julian Schick
Technische Universität Berlin
Institute of Mathematics/innoCampus
E-mail: schick@math.tu-berlin.de

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

courses. Today, more than 80 large courses are distributing their students into more than 1,000 tutorials (for details see [1]).

Due to its great success, *Moses* has been extended to solve the examination timetabling problem. Since 2015, *Moses* has also been used at the Technical University of Munich (more than 39,000 students) for the examination timetabling problem (for details see [2]).

For 2013, RWTH Aachen University was expecting a larger number of first-year students. At the same time, a delay was expected in the construction of a new lecture hall at the university. Therefore, in 2011, RWTH Aachen University asked *innoCampus* for support in improving the coordination of their resource "room". In 2012, *innoCampus* and the department of operations research at RWTH Aachen University started the *carpe diem project*. This resulted in the extension of *Moses* to solve the university course timetabling problem for RWTH Aachen University. This involved more than 42,000 students, 1,200 lecturers, and 500 rooms. One-and-a-half years later, TU Berlin also stopped copying the old course timetables and started to create them automatically using Moses [3].

2 Approach

Our approach to solving the timetabling problem 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 that for a successful optimization in the real world, this workflow is as important as a carefully designed optimization algorithm.

3 Process

On inspecting the whole planning process, we note that two important user roles are involved: On the one hand, there are normal users—e. g., lecturers, administrative employees and students. On the other hand, we have supervisors managing the timetabling process. We can identify the following six steps in the process:

1. Preparation (supervisors)
2. Data input (users)
3. Data cleaning (supervisors)
4. Timetable creation (supervisors)
5. Internal publication and review of the timetable (supervisors/special users)
6. Publication of the timetable (supervisors)

What are the key properties of the system that facilitate this process significantly? For each task in this planning workflow, the system provides clear user interface pages that are rich in explanations and hints. The software guides

users through the planning process. Moreover, data quality is encouraged and even enforced on the data input. This is ensured through hints in the user interface and the system's refusal to store incomplete or invalid datasets. The need for supervisors to correct the data and carry out further inquiries is thus minimized. In case the supervisors still need to do troubleshooting, they can rely on the tools built into the system. Specialized statistics and overviews allow them to find structures in the data that are likely to be incorrect or, heuristically, to cause trouble in the optimization step. Consider the following scenario: Several teachers are listed for a certain course. The total amount of time every week during which these teachers are free to lecture is, say, five hours. But the lecture runs for six hours per week. Conflicts like this can be identified by the built-in tools. The optimization itself can easily be monitored and triggered within the graphical user interface, so that there is no need for console work and manual data manipulation.

Once a schedule has been calculated, the work is not done yet. Diverse views and manipulation methods present the plan in a clear fashion to the university employees and enable the supervisors to correct or update the schedule where required, in consultation with the stakeholders. Even normal users can adjust the schedule providing no conflicts are produced by their actions.

Finally, a flexible and powerful user permission management plays a vital role in controlling the workflow outlined above. The organizational structure of most universities is 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.

4 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, it is solved using a commercial MIP solver.

Only our careful design of the model guarantees that even large timetabling problem instances lead to an acceptable solution (less than 20 % from optimum) within an acceptable time span (within hours; see [3] for detailed statistics). For instance, preprocessing is carried out to reduce the number of MIP constraints. The constraints that ensure every course is assigned to a timespan and a room are soft constraints; therefore, infeasible models are avoided. An infeasible model would yield no information to planners, whereas a solution with a few unassigned courses shows at which point looser input constraints could be necessary.

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 can tell exactly how far from the optimum we are in the worst case.

5 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, the 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.

6 Future Work

The bigger the system grew, the more obvious it became that modularization is necessary. Decomposition into pluggable modules, therefore, is 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 is that the system could be better adapted to the specific needs of the universities employing 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)