Management and Optimal Distribution of Large Student Numbers

Wojtek Gora, Gerald Lach, Jan Lübbe,
Olivier Pfeiffer, Erhard Zorn
Center for Multimedia in Education and Research
Technische Universität Berlin
Berlin, Germany
{gora, lach, mlach, luebbe, pfeiffer, erhard}@math.tu-berlin.de

Sabina Jeschke
Faculty of Mechanical Engineering
RWTH Aachen University
Aachen, Germany
sabina.jeschke@zlw-ima.rwth-aachen.de

Abstract— **In principle, timetabling problems appear at every school and university. However, the degree of difficulty increases dramatically with an increasing number of students and courses for which the scheduling shall be carried out. From the mathematical point of view this is a "hard" problem, since the runtime on a computer cannot be estimated by a simple law (i.e. by a polynomial law) by the number of parameters. These kinds of problems are called "NP hard". There are three important versions of the timetabling problem at universities, known as the university timetabling problem, i.e. curriculum-based course timetabling, post-enrollment timetabling and examination timetabling.**

All specified problems are important for room management at universities, for the realization of courses that can be studied according to curricula, and for the satisfaction of students and teachers. These problems are related to the optimization of room management and personnel costs (e.g. by a uniform distribution of students). Thus, the solution to these problems is related to the optimization of "real" costs, a more and more important economic factor at (German) universities. Introduction of the two-tiered Bachelor and Master courses has raised awareness for these problems at German universities: due to the multitude of new courses the timetables which have been established and stood the test of time cannot be used any longer. Moreover classes tend to be more structured and have strong similarities to classical school situations; attendance is compulsory and dependencies between modules determine the feasibility of the curricula. This feasibility is also evaluated while accrediting new study courses. Since 2003, we have been using using an algorithm that has been realized by members of our team for the solution of the post-enrollment-based course timetabling problem at the Technische Universität Berlin. .

For classes with more than 2000 enrolled students, organization by itself is a challenge; problems may include splitting of those classes into several separate lectures, arranging the associated tutorials into small groups of students, allocating adequate rooms, and scheduling examinations. Moreover, homework and exams need to be administered, whereby, depending on the field of study, very different rules are to be obeyed. This especially pertains to the Department of Mathematics because it offers most of the compulsory classes in mathematics for all fields of study held at Technische Universität Berlin. These are the biggest classes at the university and are to be attended by the majority of students. Thus, the Moses (Mobile Services for Students)-Account has been being developed and used since 2004. This web-based software allows students to enroll in tutorials with a list of preferences for given dates. A special algorithm, providing a globally optimized (with respect to the students' wishes and resources available) solution, processes all registrations.

Keywords—university timetabling, academic administration, integer programming, NP-completeness

I. INTRODUCTION

One of the major challenges facing universities is the organization of the study supporting processes, especially for first year students. With about 28.000 students, TU Berlin is one of the German universities of technology with the largest student body. As a service for other schools, the Department of Mathematics is in charge of the mathematical education of most students, irrespective of their actual course of studies, making it the biggest „service provider" of the university. Since a major reorganization of the math-service seven years ago, every year, several thousand students from 18 different courses of studies are participating in one of the six „math-service" engineering modules (Analysis I-III, Differential Equations, Integral Transformations & Partial Differential Equations, Linear Algebra), altogether filling 9.000 to 10.000 spots in the above-mentioned modules. The largest class is the freshman course in Linear Algebra with a total count of more than 2.200 students per semester. The modules are organized as follows: In order to guarantee lecture class sizes of less than 350 attendees, one and the same lecture is being held numerous times throughout the week, sometimes even at the same time. The lecturers of the individual classes take care of presenting the same mathematical material to all attendees. In addition to lecture classes, all students have to solve the same exercises as homework and, in the case of passing the homework assignments are admitted to take the written examination at the end of the semester. Students also are eligible to attend small-sized recitation/exercise classes (so-called tutorials) in which the material of the lectures is manifested. The following list summarizes the administrational duties for performing these classes:

978-1-4244-6571-2/10/$26.00 ©2010 IEEE

- Assigning all students to exercise classes
- Managing homework scores for admission to final examinations and course credits
- Managing registration of students for examinations
- Informing students about examination scores
- Collecting and submitting examination scores to the university's central examination office

Although all of the above tasks are typical administrational duties for student management at universities, the large course sizes make them very time-consuming and labor-intensive. This is especially true for the assignment into exercise groups for every module: All students have to be assigned to small (meaning total sizes of 15 to 30 students each) groups/tutorials in a way that assignments neither conflict with each other nor conflict with the individual schedules of the students while respecting certain capacity restrictions. However, the large course sizes not only present difficulties; they also imply a great opportunity of rationalization in the student administration. Against this background the development of MosesKonto (acronym for Mobile Services for Students and Konto as the German word for account) began in 2002 and was initially deployed in 2003 and has been used ever since — first for all math-service classes described above. Since the fall semester 2005/6, the assignment into exercise classes and management of exams has been extended to cover further courses, even across different schools. In 2002 we started with 5 math-services classes, in the fall semester 2009/10 54 classes offered by six of seven schools of TU Berlin are using the MosesKonto to create conflict-free timetables for their students.

II. TIMETABLING

Timetabling problems have always been an important administrative issue at universities. Due to the Bologna process [1] the construction of timetables has become more and more complicated, mainly caused by the diversity of the newly created courses of study. Consequently the need for IT-aided systems has grown. Unsurprisingly a huge amount of literature has been published over the last years ([2], [3], [4], [5], [6]). The university timetabling problems have been grouped into three parts: the university course timetabling [7], the post-enrollment timetabling [8] and the examination timetabling problem [9] . Focusing on these three problems within the scope of the PATAT 2008 [10], [11], [12], [13], an international timetabling competition (ITC2007 [14]) for solving standardized problem formulation was carried out.

A. Problem definition

The problem we focus on is related to the post-enrollment timetabling problem, which occurs for big lectures (typically with several hundreds of students) for which several small exercise/recitation classes are offered. These so-called tutorials are offered at different times and the students have to be distributed into these tutorials taking into account their individual constraints, e.g. students are enrolled in different lectures with accompanying tutorials. This means that many students with individual time constraints have to be matched conflict-free to one or more tutorials with different time slots. In general these constraints are known as "hard constraints", i.e., a timetable violating some of them is not suitable. A timetable satisfying all the hard constraints is called a feasible timetable. Furthermore the students are able to give a rating for the time slots (often called "wishes" or "soft constraints"). The goal of our solution approach is to find a feasible timetable which meets the student ratings as closely as possible. For us this seems to be an appropriate procedure for the post-enrollment course timetabling problem at a huge German university. Comparing our formulation to the one of ITC2007 we have to note that the hard constraints are both identical, but in the soft constraint definition the formulations differ a lot. In our opinion, solving our problem formulation (using our soft constraints) is easier than optimizing the problem formulation of the ITC2007. However, when comparing the problem sizes, we see that our problem with more than 5.000 participating students, more than 700 events and more than 17.000 tutorial-seats is much bigger than the problems of ITC2007 [15].

At the beginning of the project in 2002, we looked for a commercial software tackling our specific timetabling problem. To the best of our knowledge no suitable software tool satisfying our requirements was available at that time. By now several providers of timetabling specific software solutions exist, commercial ones as Scientia [16], CELCAT [17], Mimosa [18] or EMS [19], and non-commercial ones, as UniTime [20]. However, we believe that none of them can either cope with such huge problem instances like ours or do they have the same problem definition.

B. Solution method and comparison

The optimization step involves the computation of all assignments of students into tutorials, in a way that room and teaching staff capacities are respected and the computed solution is optimal in respect to the chosen priorities of dates. The problem admits a formulation as a constrained minimum-cost-flow network problem (Fig. 1, for details see [21]).

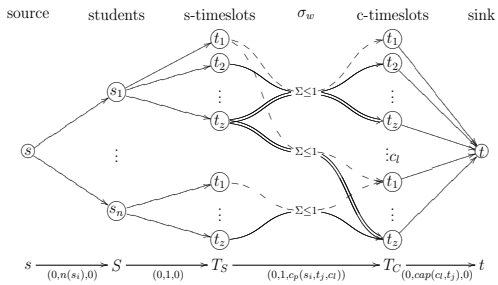

Figure 1. Network for several courses (boundaries and costs for all edges are denoted symbolically in the lower row)

To describe this in terms of a network flow problem let $s \in S$ be a student. Nodes $t_j \in T_S$, representing distinct time slots, are linked to the student and each link of such a node to another $t_j \in T_C$ represents a rating. Each $t_j \in T_C$ represents one or more tutorial of the course C at this time slot, and $cap(t_j,c_l)$ defines the capacity of this tutorial for this time slot as the upper bound of the arc linking the tutorials with the sink. The total demand

for tutorials is defined as the flow at the source s. To safeguard that each student is assigned exactly to one seat in a tutorial for each course, some bundle cuts w are defined, forcing this property.

The resulting integer linear program finds a solution in acceptable time, i.e. within less than one minute for the largest instances seen so far on a recent machine. The data of the MosesKonto are accessed and preprocessed by the software TUTOP [22], which also formulates the integer program that is then to be solved by the commercial software CPLEX [23].

$$\min \sum_{s,c,ts} prio(s,ts) \cdot y_{s,c,ts} \tag{1}$$

$$\text{s.t.} \quad \sum_{ts \in \mathcal{T}} x_{s,ts} = \ell(s) \qquad \forall s \in \mathcal{S} \tag{2}$$

$$x_{s,ts} - \sum_{c \in \mathcal{C}(s)} y_{s,c,ts} = 0 \qquad \forall (s,ts) \in \mathcal{S} \times \mathcal{T} \tag{3}$$

$$\sum_{s \in \mathcal{S}(c)} y_{s,c,ts} \leq cap(c,ts) \qquad \forall (c,ts) \in \mathcal{C} \times \mathcal{T} \tag{4}$$

$$\sum_{ts \in \mathcal{T}} y_{s,c,ts} \leq 1 \qquad \forall (s,c) \in \mathcal{S} \times \mathcal{C} \tag{5}$$

$$x_{s,ts}, y_{s,c,ts} \in \{0,1\} \tag{6}$$

Figure 2. Resulting integer linear program

The integer program (1)–(6) includes two kinds of binary variables – $x_{s,ts}$ and $y_{s,c,ts}$. In a solution a variable $x_{s,ts}$ equals one if and only if student s Є S has been assigned a tutorial at timeslot t_s otherwise $x_{s,ts}$ is zero. A variable $y_{s,c,ts}$ is set to one if student s Є S has been assigned a tutorial of the course c at timeslot t_s. Based on these variables we are able to formulate linear constraints specifying our version of the post-enrollment timetabling problem. In (1) the objective function is described, thereby *prio*(s,t_s) reflects the priority of the student s to have a tutorial at timeslot t_s. The goal of this linear integer program is to find a variable setting of the $x_{s,ts}$, $y_{s,c,ts}$ variables satisfying the constraints (2)–(6) and minimizing the objective function (1). The constraints (2) affirm that every student gets assigned to as many tutorials as he/she applied for. In (3) the two kinds of variables are linked. That means if the variable $x_{s,ts}$ equals one, or in other words, if the student s has been assigned some tutorial at timeslot t_s, there must be exactly one course c where student s participates in a tutorial at this timeslot t_s. To put it differently, if the student s has no tutorial at timeslot t_s (the variable $x_{s,ts}$ equals zero), he cannot be in any tutorial of other courses at this timeslot, and in consequence all the variables $y_{s,c,ts}$ must be zero. The constraints (4) safeguard that the capacity of the different courses are never exceeded. Finally in (5) the earlier mentioned bundled cuts are specified. The last constraints (6) postulate that all variables should have the values one or zero in a feasible solution.

Many problem-solution approaches of the post-enrollment timetabling problem use heuristic procedures ([6], [24], [20], [6], [25], [26]). Generally these techniques are able to provide good solutions but they are not capable to prove optimality, i.e. the user never knows if the current solution is the best one or if there exists a better one. On the contrary, approaches based on integer programming as ours are exact ([27], [28]). This means, if the problem is solved, the user can be sure that the found solution is a global optimum. This seems to be a very important feature for all users of the software. Several exact methods for solving different timetabling problems are known ([29], [30], [31], [32]), but to the best of our knowledge none of them is concerned with the post-enrollment timetabling problem.

C. Statistics

In table 1 some statistics concerning the development of the TUTOP specific key data are presented. We can easily see that the total number of participating users/courses has been increasing enormously for the last few years. This becomes even more apparent when we are considering the fall and spring semester separately. Another observation is that the number of tutorial spots has tripled since 2004, whereas the number of students has just doubled. This is due to the fact that for every student more courses of his current timetable are involved. In the fall semester 2004/05 every student applied for 1.65 tutorial spots on average and today is applying for 2.66 tutorial spots. Taking into account that we affirm the conflict-freeness for every student's timetable and caring for his personal preferences, this seems to be a surplus for every single student. The more courses are participating the more the students profit from their personalized automatic timetable creation resulting in a high satisfaction of the students and course offerers. The rate of approval for our work has significantly increased over the last years.

TABLE I. OVERVIEW OF TUTOP KEY DATA STATISTICS OF THE LAST FIVE YEARS

	courses	**students**	**tutorials**	**spots**	**Ø priority**	**tutorials/courses**
winter semester 2004/05	5	3270	163	5392	1.24	1.65
summer semester 2005	6	2681	149	4197	1.27	1.57
winter semester 2005/06	14	4051	326	8925	1.37	2.20
summer semester 2006	17	3569	350	8120	1.40	2.28
winter semester 2006/07	24	4507	418	11626	1.58	2.58
summer semester 2007	29	4248	461	11770	1.59	2.77
winter semester 2007/08	37	5173	540	14533	1.24	2.81

Summer semester 2008	38	4854	642	14015	1.24	2.89
winter semester 2008/09	46	5472	626	15139	1.23	2.77
summer semester 2009	46	5337	807	14032	1.29	2.63
winter semester 2009/10	54	6421	765	17073	1.26	2.66

III. IMPLEMENTATION

The MosesKonto software was implemented in Java, using a JSF [33] driven graphical user interface and using the latest web technologies within the Icefaces Java application framework [34]. The web application is running on a Java GlassFish Application Server [35] using a postgreSQL database [36].

The high adaptability and reliability of the web-based MosesKonto software induced not only a high number of users, but also an increased acceptance, especially among the course providers. This development manifested in a huge number of course providers willing to work on their administrational duties and responsibilities with the MosesKonto software. However, there is a difference between the spring and the fall semester since different courses are using the MosesKonto for creating timetables for their students. Some of the participating courses are using the MosesKonto only for the administration of written examinations. The combination of over 200 different courses (with "only" 54 of them using the software for timetabling in the fall semester of 2009/10) as well as over 22.000 individuals using the MosesKonto creates new administrational and organizational challenges for the MosesKonto. One of these challenges is the allocation of the different supported administrative tools and the private user data involved, which occasionally concerns sensible data like the results of written examinations. It was necessary to implement a user rights management system on the MosesKonto not only in order to protect the privacy of the data but also as a duty impelled by the proceeding process of decentralization When the MosesKonto was started in 2003, we created our own user management due to the lack of a centralized authorization system.

Since fall semester 2007/08 we have been using the central LDAP server (Lightweight Directory Access Protocol [37]) of TU Berlin. LDAP was originally invented to make frequently-used data available faster and more easily. It is also often used to authenticate users. Every member of the university has a central tubIT account (tubIT: IT service center of the TU Berlin), with which it is possible to use several independent services (e.g. email, WLAN), always using the same username and password.

Although single sign on frameworks such as Kerberos are used, one is not handed from one service to another service. Instead one needs to enter the username and password for each service separately. Therefore LDAP works much better than Kerberos and for a great number of applications standard LDAP modules exist. Nevertheless only one single central user administration is needed for different services.

In the near future, the MosesKonto will be integrated into the TU web portal to implement a real single sign on (with several other services offered by tubIT). Additionally, users have to be equipped with different detailed rights, depending on their role. Typically, the users of a system fall into one of the following groups requiring fundamentally differing access rights that can even vary between departments:

- Students, as a rule, are granted access to their own personal data and nothing else (exception: courses allowing team work). They are given write access for their own data and for registering for exams or exercise classes, otherwise read access only.
- The Administrator is given access to all information; he is provided with additional, special rights, such as the creation of new modules.
- Employees of the Service Center are usually responsible for all communication with the central office of examinations. The Service Center is the central contact for all students.
- Teaching Assistants are often responsible for entering the results of exams and similar tasks.
- Tutors will enter their input if the homework-related criteria are met and have access to the personal data of their students.

The growing interest of other schools in using the MosesKonto for the registration of their own exercise classes, examination and student administration outside the mathematical service modules required an expansion of the access right management to facilitate the independent administration of their students. These changes are responsible for different access right requirements for the above mentioned user groups. As a solution, we have implemented a hierarchical access rights management system mirroring that of UNIX [38]. All reading or writing rights for the different types of information and user interactions (e.g. access to the students' personal data or to their examination results) are treated as a single, independent resource, whereas only the writing rights allow changing or submitting data. Passing these resources (reading or writing) is treated as a third type of resource. User groups (e.g. teaching assistants) can be created and are defined by their associated access rights. The owner of a group (either because he created it or it was associated with him by default) has the right to add or remove members of the group or grant access rights to group members within the limits of his own rights. Students are outside this access rights management, as their rights do not have to be defined on a per-module basis. The right to register for an exam, for example, can be managed through the use of a registration time period.

Facing the high number of courses using the MosesKonto for administrational tasks in students' management and therefore rising complexity and, more importantly, for the purpose of decentralizing the administrational duties it was necessary to develop a user-friendly and intuitive user interface for creating new resources and managing the already existing rights respectively. Therefore we created a JSF [33] driven graphical user interface using the latest web technologies within the Icefaces Java application framework [34].

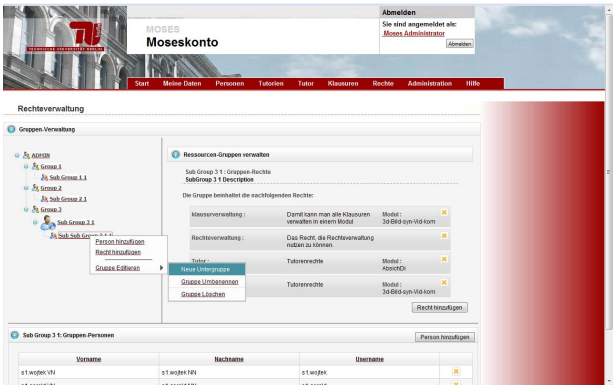

Figure 3. Tree structure of access right management

In order to be fully compatible with the webpages created in the earlier period of the MosesKonto and to keep the proven flexibility of right assignments, we stuck to the UNIX-based rights management and extended the existing three types of resources by adding a fourth type of resource, which gives the group associated to it the right of passing not only the reading or writing resource but also the right to pass this third type of resource, including the ability of passing itself. Finally, it is now possible to hand in every single task, assigned to a right, to a group of users, which again can assign it to another group or handle it themselves. Through this, the base for decentralizing the administrational tasks was created. As a next step we adopted the hierarchical structure of the rights management by arrangement of the groups in a tree structure.

If a member of an existing right group wants to create a new group now, he or she can only create it as a subgroup of the original group and pass the rights within the limits of his own rights. As a result, the tree structure develops on its own.

The new tree-structured user interface also simplifies the allocation of rights to a greater number of groups by associating all upper groups with rights associated to lower groups and vice versa. The upper groups are also able to delete or change the rights of the lower groups.

As a last step in improving the rights management system of the MosesKonto we combined the four types of resources mentioned before to a coherent group of resources.

This new created "resource-group" forms a layer of rights which, from the user's point of view, appears as an independent package of rights and can be assigned to different groups whereas the content, or more precisely the included resources, can only be controlled by the administrator.

However, one should bear in mind that these "resource-groups" are not used for verification of the given rights, but only the resources within the groups are relevant for the purpose of usage.

On the one hand this centralization of resources indeed decreases the flexibility of right allocation for the user, but on the other hand it increases the acceptance and usability of our right management system by letting the administrator combine sensible resources and give them meaningful and significant names and descriptions.

IV. Outlook

The rapidly increasing number of courses using our tool motivated our group to develop more software solutions for different timetabling problems. We have recently finished the implementation of a tool which creates examination timetables for a huge number of written exams affirming that the students have enough time for preparation between two exams [39].

References

[1] The Bologna Declaration on the European space for higher education: an explanation, document prepared by the Confederation of EU Rectors' Conferences and the Association of European Universites (CRE), available at (last accessed 21-January-2010) http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf .

[2] E. Burke, K. Jackson, J.H. Kingston, and R. Weare. Automated university timetabling: The state of the art. Comput. J., 40(9):565–571, 1997.

[3] D. de Werra. The combinatorics of timetabling. European J. Oper. Res., 96:504–513, 1997.

[4] A. Schaerf. A survey of automated timetabling. Artificial Intelligence Review, 13(2):87–127, 1999.

[5] E.K. Burke and S. Petrovic. Recent research directions in automated timetabling. European J. Oper. Res., 140(2):266–280, 2002.

[6] R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum, 30(1):167–190, 2007.

[7] Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). In Federico Pecora and Nicola Policella, editors, Proceedings of the 1st International Workshop on Scheduling a Scheduling Competition (SSC 2007), Providence (RI), USA, September 22–27 2007.

[8] R. Lewis, B. Paechter, and B. McCollum. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff Working Papers in Accounting and Finance A2007-3, Cardiff Business School, Cardiff University, August 2007.

[9] Barry McCollum, Paul McMullan, Edmund K. Burke, Andrew J. Parkes, and Rong Qu. The second International Timetabling Competition: Examination timetabling track. Technical Report QUB/IEEE/Tech/ITC2007/Exam/v4.0/17, Queen's University, Belfast, Sep 2007. Available from http://www.cs.qub.ac.uk/itc2007/.

[10] E.K. Burke and H. Rudova, editors. PATAT 2006: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Lect. Notes Comp. Science, Berlin, 2007. Springer. To appear; available online at http://patat06.muni.cz/proceedings.html.

[11] E.K. Burke and M. Gendreau, editors. PATAT 2008: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, Annals of Operations Research, Berlin, 2009. Springer. To appear; available online at http://w1.cirrelt.ca/~patat2008/PATAT_7_PROCEEDINGS/Papers/Papers.htm.

[12] PATAT: Practice and Theory of Automated Timetabling. http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml.

[13] PATAT08: Practice and Theory of Automated Timetabling 2008. https://symposia.cirrelt.ca/PATAT2008/en.

[14] ITC2007: 2nd International Timetabling Competition. http://www.cs.qub.ac.uk/itc2007/.

[15] R. Lewis. A time-dependent metaheuristic algorithm for post enrolment-based course timetabling. In Proceedings of PATAT 2008, 2008.

[16] Scientia. http://www.scientia.com/uk/.

[17] CELCAT. http://www.celcat.com/.

[18] Mimosa. http://www.mimosasoftware.com/.

[19] EMS Software for Educational Facilities. http://www.dea.com/Industries/Education/Default.aspx.

[20] UniTime. http://www.unitime.org/.

[21] D. Jungnickel. Graphs, Networks and Algorithms. Springer, 1998.

[22] R. Luce. Tutop: Tutorienplaetze optimal verteilen. Master's thesis, Technische Universität Berlin, Institut für Mathematik, 2003. In German.

[23] Ilog cplex. http://www.ilog.com/products/cplex/.

[24] C. Meyers and J.B. Orlin. Very large-scale neighborhood search techniques in timetabling problems. In PATAT 2006: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, pages 36–52.

[25] Zhipeng Lu and Jin-Kao Hao. Solving the course timetabling problem with a hybrid heuristic algorithm. LNAI, 5253:262–273, 2008. Proceedings of 13th International Conference on Arti?cial Intelligence: Methodology, Systems, Applications (AIMSA'08). D. Dochev, M. Pistore, and P. Traverso (eds).

[26] Tomas Müller. ITC2007 solver description: a hybrid approach. Annals of Operations Research.

[27] A. Schrijver. Combinatorial Optimization Polyhedra and Efficiency. Springer, Berlin, 2003.

[28] A. Schrijver. Theory of Linear and Integer Progamming. Wiley, J, 1998.

[29] S. Daskalaki and T. Birbas. Efficient solutions for a university timetabling problem through integer programming. European J. Oper. Res., 127(1):106–120, January 2005.

[30] A. Qualizza and P. Serafini. A column generation scheme for faculty timetabling. In E.K. Burke and M.A. Trick, editors, PATAT 2004: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling, volume 3616 of Lect. Notes Comp. Science, pages 161–173, Berlin, 2005. Springer.

[31] Edmund K. Burke, Jakub Marecek, Andrew J. Parkes, and Hana Rudova. A branch-and-cut procedure for the udine course timetabling problem. In Edmund K. Burke and Michel Gendreau, editors, Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008, Montreal, CA, 2008. Universite de Montreal. Also appeared as NOTTCS-TR-2008-1.

[32] G. Lach and M.E. Lübbecke. Optimal university course timetables and the partial transversal polytope. In C.C. McGeoch, editor, 7th International Workshop on Efficient and Experimental Algorithms (WEA08), volume 5038 of LNCS, pages 235–248, Berlin, 2008. Springer.

[33] JavaServer Faces Technology. http://java.sun.com/javaee/javaserverfaces/.

[34] IceFace. http://www.icefaces.org/main/home/.

[35] GlassFish project. https://glassfish.dev.java.net/.

[36] Postgesql. http://www.postgresql.org/.

[37] Timothy A. Howes, Mark C. Smith, and Gordon S. Good. Understanding and Deploying LDAP Directory Services. Addison-Wesley Professional, 2 edition, May 2003.

[38] Brian W. Kernighan and Rob Pike. The UNIX programming environment. Prentice-Hall Software series. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[39] M. Lach. Ein Verfahren zur Optimierung der Klausurterminplanung an der TU Berlin. Master's thesis, Technische Universität Berlin, Institut für Mathematik, 2008. In German.