Curriculum Based Course Timetabling: Optimal Solutions to the Udine Benchmark Instances

Gerald Lach* Marco E. Lübbecke*

February 27, 2008

Abstract

We present an integer programming approach to the university course timetabling problem, in which weekly lectures have to be scheduled and assigned to rooms. Students' curricula impose restrictions as to which courses may be scheduled in parallel. Besides some hard constraints (no two courses in the same room at the same time, etc.), there are several soft constraints in practice which give a convenient structure to timetables; these should be met as well as possible.

We report on solving benchmark instances from two International Timetabling Competitions which are based on real data from the university of Udine. The first set is solved to proven optimality, for the second set we give solutions which do not violate any hard constraints. We further present solutions to larger instances with more elaborate hard constraints from TU Berlin, Germany.

Keywords: Integer Programming; Decomposition; Timetabling

1 Introduction

Curriculum based course timetabling is to assign weekly lectures to time periods and rooms in such a way that a number of hard constraints are fulfilled (if this is impossible, the number of violations is to be minimized), and several soft constraints are best possibly met. The problem, also known as university course timetabling, received much attention in the operations research literature, see the surveys [3, 11], not least due to the fact that practical data is available for benchmarking, in particular instances from the university of Udine [7], which came from ITC2002, the first International Timetabling Competition. In ITC2007, the second issue of the competition (`www.cs.qub.ac.uk/itc2007`), more benchmarks from Udine were provided, together with extended problem definitions, in particular for the soft constraints.

In this paper, we report for the first time on solving the four original (2002) Udine instances to proven optimality, and give solutions which do not violate any hard constraint to the 2007 instances. For some of them we obtain optima. We furthermore provide solutions to instances from Berlin's Technical University which feature slightly more elaborate hard constraints.

We approach the problem (which is NP-hard) via integer programming, as has been proposed before, see e.g., [2, 4, 5, 6, 10, 12]. However, instead of directly solving a natural formulation

*Technische Universität Berlin, Institut für Mathematik, MA 5-1, Straße des 17. Juni 136, 10623 Berlin, Germany, Email: {`lach, m.luebbecke`}`@math.tu-berlin.de`.

based on three-indexed variables for the course/time/room assignment, we decompose the problem in two stages. In a first step, we only match time periods and lectures; these pairs are then feasibly assigned to rooms in a second step. This decomposition is exact with respect to hard constraints, that is, no solutions are lost. This can be achieved by implicitly taking care about feasibility for room assignments already in the first stage. Overall, this approach results in easier integer programs, and thus larger instances to be solved.

University Course Timetabling

Each course consists of several lectures, each day consist of several time slots. A (day, time slot) pair is called a period. A curriculum is a set of courses no two of which may be scheduled in parallel. Every lecture has to be scheduled to a period in a room which provides enough seats to host the lecture (in our Berlin instances, each room must also provide the requested features like beamer, PC, blackboard, location, etc.). No two courses by the same teacher or which appear in the same curriculum may be scheduled at the same period; no two courses may take place at the same period in the same room. All these constraints are considered *hard*. We defer the subtleties of soft constraints to our discussion on how to formulate them in our integer programs.

In a companion paper [8] we developed the theoretical background for the decomposition which considered hard constraints only. In this paper, we report on how to make it useful in practice, in particular, we state how to incorporate a variety of soft constraints.

2 The Hard Constraint Solver Framework

Our focus is on keeping all hard constraints (resp. as many as possible); thus, the core of our model is built around this goal. Soft constraints are added as needed; see Section 3.

Denote by $\mathcal{C}$ the set of courses, by $\mathcal{R}$ the set of rooms, and by $\mathcal{P}$ the set of periods. For each course $c \in \mathcal{C}$ we know its eligible periods $P(c) \subseteq \mathcal{P}$, eligible rooms $R(c) \subseteq \mathcal{R}$, and that $\ell(c)$ lectures have to be scheduled; that is, we have to provide $\ell(c)$ different periods for course c. As an example objective function we formulate teachers' preferences $prio(c, p)$ for course/period combinations; the smaller the number, the higher the priority.

Time conflicts of any kind are represented via a conflict graph $G_{\text{conf}} = (V_{\text{conf}}, E_{\text{conf}})$: A vertex (c, p) represents an eligible combination of course c and period p. Two nodes (c_1, p_1) and (c_2, p_2) are adjacent iff it is forbidden that c_1 is scheduled at p_1 and c_2 is scheduled at p_2 (typically, $p_1 = p_2$). This stable set characteristic of the problem motivated several researchers to relate timetabling to graph coloring, see e.g., [2], and references therein.

Instead of using binary variables which represent whether course c is scheduled at period p in room r, we reduce the problem in three dimensions to a problem in two dimensions, implicitly taking care of room conflicts. To this end, we represent eligible combinations of courses and rooms as undirected bipartite graphs $G_p = (\mathcal{C}_p \cup \mathcal{R}_p, E_p)$, one for every period $p \in \mathcal{P}$. Courses which may be scheduled at p are given in set $\mathcal{C}_p$, and $\mathcal{R}_p$ denotes the set of all eligible rooms for all courses in $\mathcal{C}_p$. A course c and a room r are adjacent iff r is eligible for c. For ease of exposition let $G = (\mathcal{C} \cup \mathcal{R}, E)$ be the graph consisting of all components G_p, $p \in \mathcal{P}$.

For a subset $U \subseteq \mathcal{C}$ of vertices, denote by $\Gamma(U) := \{i \in \mathcal{R} \mid j \in U, (i, j) \in E\}$ the neighborhood

of U. Hall's stable marriage theorem [9] states that a bipartite graph $G = (\mathcal{C} \cup \mathcal{R}, E)$ has a matching of all vertices in $\mathcal{C}$ into $\mathcal{R}$ if and only if $|\Gamma(U)| \geq |U|$ for all $U \subseteq \mathcal{C}$. Enforcing this condition, we are able to schedule courses in such a way that rooms can be assigned later.

We call this the *first stage of the decomposition*. The resulting integer program obviously has an exponential number of constraints (3), and we give details in [8] on how to cope with this.

$$\min \sum_{(c,p)\in V_{\text{conf}}} prio(c,p) \cdot x_{c,p} \tag{1}$$

$$\text{s.t.} \sum_{p\in P(c)} x_{c,p} = \ell(c) \qquad \forall c \in \mathcal{C} \tag{2}$$

$$\sum_{c\in U} x_{c,p} \leq |\Gamma(U)| \qquad \forall U \subseteq \mathcal{C},\ p \in \mathcal{P} \tag{3}$$

$$x_{c_1,p_1} + x_{c_2,p_2} \leq 1 \qquad \forall ((c_1,p_1),(c_2,p_2)) \in E_{\text{conf}} \tag{4}$$

$$x_{c,p} \in \{0,1\} \qquad \forall (c,p) \in V_{\text{conf}} \tag{5}$$

Once this program is solved, the *second stage of the decomposition* merely consists of solving a sequence of minimum weight bipartite perfect matching problems in polynomial time, one for each period. Clearly, this decomposition approach is exact.

3 Integrating Soft Constraints

Besides mandatory constraints there is a wealth of possibilities for constraints which cannot be kept in general, but best possibly fulfilling them gives desired structures to timetables. For these soft constraints, we stick to the definitions from ITC2007, see again [7]. Four types are mentioned (and defined below): *RoomCapacity (RC), MinimumWorkingDays (MWD), CurriculumCompactness (CC), and RoomStability (RS)*. The first three can easily be included in the first stage of the decomposition. On the other hand, the RS constraints need to go in the second stage, and are ignored in the first. As a consequence, we theoretically may miss a globally optimal solution, even when both stages are optimally solved. However, in that case, solution quality would not significantly decrease since the RS constraints are the least important soft constraints. Penalties for violations are taken from [7].

3.1 RoomCapacity Constraints

A room should provide as many seats as requested by each assigned course. A penalty occurs for each missing seat. This constraint is a hard constraint in our original framework; here, however, we treat is as soft. One might expect to handle room capacity in the second stage of the decomposition, but a modification of Hall's conditions (3) already does the job.

Let p be an arbitrary but fixed period. Constraints conditions (3) are replaced by the following set of constraints. We first require the number of courses that can take place at p to be at most the number of available rooms:

$$\sum_{c\in\mathcal{C}} x_{c,p} \leq |\mathcal{R}| \ . \tag{6}$$

This avoids conflicts in the assignment of rooms. Next, we introduce constraints that take the different room capacities and demands of the courses into account. Denote by $\mathcal{S}$ be the different room capacities. Let $\mathcal{C}_{\geq s}$ denote all courses with demand larger than s; and $\mathcal{R}_{\geq s}$ denotes rooms with capacity more than s seats. For each $s \in \mathcal{S}$, except the smallest, and for all $c \in \mathcal{C}_{\geq s}$ there is a binary variable $y_{s,c,p}$. We add

$$x_{c,p} - y_{s,c,p} \geq 0 \qquad \forall s \in \mathcal{S},\ c \in \mathcal{C}_{\geq s} \tag{7}$$

$$\sum_{c \in \mathcal{C}_{\geq s}} x_{c,p} - y_{s,c,p} \leq |\mathcal{R}_{\geq s}| \ . \tag{8}$$

Variable $y_{s,c,p}$ is set to one if course c takes place in a room of capacity smaller than s. By constraint (8) we ensure that this does not happen for more courses than we have rooms of appropriate capacity; otherwise, we incur a penalty which is considered in the objective function. Variable $y_{s,c,p}$ receives the coefficient $obj_{s,c,p}$ which reflects the difference between the demand of course c and s. We add to the objective function (1)

$$\sum_{c \in \mathcal{C}_{\geq s}} obj_{s,c,p} \cdot y_{s,c,p} \ . \tag{9}$$

3.2 MinimumWorkingDay Constraints

For each course c we specify a minimum number $mnd(c)$ of days, among which its lectures should be distributed. This constraint goes into the first decomposition stage. We introduce a binary variable $z_{c,d}$ for every course c and every eligible day d for this course. Now we add

$$\sum_{p \in d} x_{c,p} - z_{c,d} \geq 0 \qquad \forall c \in \mathcal{C},\ d \in \mathcal{D} \ . \tag{10}$$

So, $z_{c,d}$ can be set to one only if course c takes place at some period of day d. Furthermore, we introduce another integer variable w_c and the following constraint:

$$\sum_{d \in \mathcal{D}} z_{c,d} + w_c \geq mnd(c) \qquad \forall c \in \mathcal{C} \tag{11}$$

Obviously, variable w_c may take value zero only if course c takes place on more than $mnd(c)-1$ days. According to the penalty system introduced in [7] we add to the objective function (1)

$$\sum_{c \in \mathcal{C}} 5 \cdot w_c \ . \tag{12}$$

3.3 CurriculumCompactness Constraints

For every curriculum, the corresponding courses should take place consecutively over a day. We will see that, even though easily incorporated in the first stage, these soft constraints have a negative influence on solution times. For every period $p \in \mathcal{P}$ and every curriculum $cu \in \mathcal{CU}$ we introduce a binary variable $r_{p,cu}$ and the following constraint:

$$\sum_{c \in cu} x_{c,p} - r_{cu,p} = 0 \qquad \forall cu \in \mathcal{CU},\ p \in \mathcal{P} \tag{13}$$

Variable $r_{cu,p}$ assumes value one if some course of curriculum cu takes place at period p, and zero otherwise. Note that constraints (13) imply the stable set conditions (4). Again with the help of binary indicator variables $v_{cu,p}$ we model curriculum compactness:

$$-r_{cu,p-1} + r_{cu,p} - r_{cu,p+1} - v_{cu,p} \leq 0 \tag{14}$$

If period p is the last of the day the term $r_{cu,p+1}$ is omitted, and if p is the first period of the day the term $r_{cu,p-1}$ is omitted. Obviously, $v_{cu,p}$ has to be set to one if the curriculum cu has an isolated lecture at period p. Consequently, the following term is added to the objective (1):

$$\sum_{cu \in \mathcal{CU}, p \in \mathcal{P}} 2 \cdot v_{cu,p} \tag{15}$$

3.4 RoomStability Constraints

Room stability encourages all lectures of a course to take place in the same room. In contrast to the previous soft constraints, we see no way to respect this already in the first stage. As a consequence, the perfect matching structure of the second stage is destroyed, in particular integrality of solutions is lost, and we have to resort to integer programming. The negative impact on running times is significant.

As will be seen in Section 3.6 the IP Formulation of the second stage still resembles the standard matching formulation on bipartite graphs. We introduce binary variables $u_{c,p}v_{r,p}$ which assume value one iff course c takes place in room r at period p. Furthermore, we add binary variables $y_{c,r}$ for each course c and each eligible room r, which are included via

$$\sum_{p \in \mathcal{P}} u_{c,p}v_{r,p} - |\mathcal{P}| \cdot y_{c,r} \leq 0 \ . \tag{16}$$

Variable $y_{c,r}$ must assume value one, if course c takes place in room r at least once. The second stage objective function reads

$$\sum_{c \in \mathcal{C}, r \in \mathcal{R}} y_{c,r} \ . \tag{17}$$

Clearly, if (17) is minimized over all feasible course/room assignments, the RS constraint is fulfilled best possible according to the underlying bipartite graph. But as we will see, the bipartite graph depends on the solution of the first decomposition stage. It is therefore possible that the obtained solution is not a globally optimal one.

3.5 IP Formulation for the First Stage

The introduction of soft constraints resulted in a significantly altered model as compared to (1)–(5), not only visibly but also in terms of combinatorial structures. It turns out that

this has a negative impact on computation times. The only constraint we did not yet take care of is that no two courses by the same teacher may be scheduled in parallel. Denote by $\mathcal{T}$ the set of teachers, and by $C(t)$ the courses given by teacher $t \in \mathcal{T}$.

$$\min \sum_{p\in\mathcal{P},s\in\mathcal{S},c\in\mathcal{C}_{\geq s}} obj_{s,c,p} \cdot y_{s,c,p} + \sum_{c\in\mathcal{C}} 5 \cdot w_c + \sum_{cu\in\mathcal{CU},p\in\mathcal{P}} 2 \cdot r_{cu,p}$$

$$\begin{aligned}
\text{subject to} \quad \sum_{p\in\mathcal{P}} x_{c,p} &= |P(c)| && \forall c \in \mathcal{C} \\
\sum_{c\in\mathcal{C}} x_{c,p} &\leq |\mathcal{R}| && \forall p \in \mathcal{P} \\
x_{c,p} - y_{s,c,p} &\geq 0 && \forall s \in \mathcal{S},\ c \in \mathcal{C}_{\geq s},\ p \in \mathcal{P} \\
\sum_{c\in\mathcal{C}_{\geq s}} x_{c,p} - y_{s,c,p} &\leq |\mathcal{R}_{\geq s}| && \forall s \in \mathcal{S},\ p \in \mathcal{P} \\
\sum_{p\subset d} x_{c,p} - z_{c,d} &\geq 0 && \forall c \in \mathcal{C},\ d \in \mathcal{D} \\
\sum_{d\in\mathcal{D}} z_{c,d} + w_c &\geq mnd(c) && \forall c \in \mathcal{C} \\
\sum_{c\in cu} x_{c,p} - r_{cu,p} &= 0 && \forall cu \in \mathcal{CU},\ p \in \mathcal{P} \\
-r_{cu,p-1} + r_{cu,p} - r_{cu,p+1} - v_{cu,p} &\leq 0 && \forall cu \in \mathcal{CU},\ p \in \mathcal{P} \\
\sum_{c\in C(t)} x_{c,p} &\leq 1 && \forall t \in \mathcal{T},\ p \in \mathcal{P} \\
x_{c,p} &\in \{0,1\} \\
y_{s,c,p} &\in \{0,1\} \\
z_{c,d} &\in \{0,1\} \\
w_c &\in \mathbb{Z}_+ \\
v_{cu,p} &\in \{0,1\}
\end{aligned}$$

3.6 IP Formulation for the Second Stage

Originally, the second stage was to solve a minimum cost perfect matching problem for each period. The situation is more involved in light of the soft constraints. Let $G = (U \cup V, E)$ be a bipartite graph with node set $U \cup V$ defined according to the values $x^*_{c,p}$ of variables $x_{c,p}$ obtained in the first stage. Let $cap(r)$ denote the capacity of room r and $dem(c)$ denote the seat demand of course c. Given a solution x^* the graph G is defined as follows:

$$\begin{aligned}
U &= \{u_{c,p} : x^*_{c,p} = 1\} \\
V &= \{v_{r,p} : r \in \mathcal{R}, p \in \mathcal{P}\} \\
E &= \begin{cases} u_{x,p}v_{r,p} & \text{if } y_{s,c,p} = 0 \text{ and } dem(c) \leq cap(r) \\ u_{x,p}v_{r,p} & \text{if } y_{s,c,p} = 1,\ dem(c) > cap(r),\ cap(r) = \max\{cap(\hat{r}) : cap(\hat{r}) < dem(c)\} \end{cases}
\end{aligned}$$

We denote for $x \in U \cup V$ by $\delta(x) = \{e \in E : \exists y \in U \cup V, e = xy \vee e = yx\}$ the *cut* of x in G. Then, the integer program for the second stage reads as

$$\min \sum_{c\in\mathcal{C},r\in\mathcal{R}} y_{c,r}$$

$$\begin{aligned}
\text{subject to} \quad \sum_{p\in\mathcal{P}} u_{c,p}v_{r,p} - |\mathcal{P}|\cdot y_{c,r} &\leq 0 \qquad \forall c\in\mathcal{C},\ r\in\mathcal{R} && (18)\\
\sum_{u_{c,p}v_{r,p}\in\delta(u_{c,p})} u_{c,p}v_{r,p} &= 1 \qquad \forall u_{c,p}\in U && (19)\\
\sum_{u_{c,p}v_{r,p}\in\delta(v_{r,p})} u_{c,p}v_{r,p} &\leq 1 \qquad \forall v_{r,p}\in V && (20)\\
u_{c,p}v_{c,p} &\in \{0,1\}\\
y_{c,r} &\in \{0,1\}
\end{aligned}$$

The constraints consist of two different parts. The RS constraints are given in (18), cf. (16). Constraints (19) and (20) are from the standard formulation of a (one-sided perfect) matching in a bipartite graph. The next constraint

$$\sum_{u_{c,p}v_{r,p}\in\delta(u_{c,p})} u_{c,p}v_{r,p} = 1 \tag{21}$$

ensures that each course gets one room assigned in a period when it takes place. Further, constraint (22) imposes that no room is occupied more than once at the same time.

$$\sum_{u_{c,p}v_{r,p}\in\delta(v_{r,p})} u_{c,p}v_{r,p} \leq 1 \tag{22}$$

4 Extensions

In [7] several more constraints are mentioned which are relevant in practice, but do not appear in the competition's problem definition for the purpose of a cleaner presentation. The authors state that "if in the future this formulation will prove to be inappropriate (e.g., too simple), some features could be reintroduced for future research." In this section we demonstrate how to incorporate all of them into our model; some experience is given in Section 5.

It is an advantage of our decomposition approach that several constraints, in particular those relating to rooms, are easily dealt with, some are even automatically satisfied. Conditions depending on the curriculum can be modeled via the $r_{cu,p}$ variables but require new constraints in the decomposition's first stage IP formulation.

4.1 Lunch Break for Students

For each curriculum cu and a day d let p_1, p_2 be the periods around noon. Then we add the following constraint:

$$r_{cu,p_1} + r_{cu,p_2} - l_{cu,d} \leq 1 \tag{23}$$

If curriculum cu cannot have a lunch break, because courses are scheduled around noon on day d, the binary variable $l_{cu,d}$ has to be set to one. This is penalized in the objective function with two units per violation.

4.2 Specific Patterns in Curriculum Compactness

This soft constraint is only sloppily defined in [7], but individually penalizing specific patterns of non-contiguous lectures of courses in a curriculum can be done by encoding them similarly to the pattern in constraint (14).

4.3 Curriculum Dependent Maximum Student Dayload

The maximal number $dload$ of courses a student should take in a given curriculum cu per day d can be softly limited in the same way as we encourage lunch breaks. Let $p_1, \ldots, p_k$ be the periods of day d, then we add a constraint

$$\sum_{i=1}^{k} r_{cu,p_i} - dl_{cu,d} \leq dload \tag{24}$$

The integer variable $dl_{cu,d}$ assumes a strictly positive value if the maximum dayload is exceeded. Every violation is penalized with four units.

4.4 Consecutiveness of Lectures

Some lectures have to be (or must no be) scheduled in consecutive periods. Two parts of the formulation need to be changed. The stable set conditions (4) based on the conflict graph can be adapted straight forwardly. It is more complicated, yet doable, to adjust Hall's conditions (3), but the discussion is too involved for the scope of this paper.

4.5 Room Unavailability

If a room is not available at some period p, this room simply does not appear in the corresponding bipartite graph G_p, and is omitted in the Hall's conditions (3) or equivalent constraints for this period.

4.6 Appropriate Room Sizes

A lecture should not take place in a too large room. This requirement is symmetric to the room capacity constraints, and is modeled in an analogous way. Again, let $\mathcal{S}$ be the set of different room capacities. For all except the largest $s \in \mathcal{S}$ we introduce further constraints. By $\mathcal{C}_{\leq s}$ we denote all courses with demand smaller than s, and by $\mathcal{R}_{\leq s}$ denote the rooms with capacity smaller than s. Given $s \in \mathcal{S}$, for all $c \in \mathcal{C}_{\leq s}$ we introduce a binary variable $t_{s,c,p}$ with meaning symmetric to variables $y_{s,c,p}$ in Subsection 3.1. We add constraints

$$x_{c,p} - t_{s,c,p} \geq 0 \qquad \forall s \in \mathcal{S}\ c \in \mathcal{C}_{\leq s} \tag{25}$$

$$\sum_{c \in \mathcal{C}_{\leq s}} x_{c,p} - t_{s,c,p} \leq |\mathcal{R}_{\leq s}| \tag{26}$$

A penalty reflecting the difference between s and the seat demand of course c is incurred for using $t_{s,c,p}$.

4.7 Complex Weights for Soft Constraint Violations

By our use of binary indicator variables for each individual violation of a soft constraint (that is, for each single curriculum, day, period, or room) we may give individual penalties, in particular depending on the number of students which take a given course.

4.8 Teacher Preferences

Teachers may express priorities reflecting when they prefer (not) to teach. This is the original objective function used e.g., at TU Berlin; we formulated this objective in Section 2.

5 Computational Results

We report on two entirely different sets of experiments. In the first, we deal with the Udine instances, both with ITC2002 and (more elaborate) ITC2007 soft constraint definitions. We will see that requiring curriculum compactness has a severe impact on running times. The second set contains instances with all constraints hard, partly significantly more complicated than the usual hard constraints. This set reflects the timetabling situation at the Technical University of Berlin. All experiments were run on a 3.4GHz Linux PC with 1GB memory; we solved integer programs with CPLEX 11.0. All reported optimality gaps were computed relative to the upper bound, i.e., as (upper bound − lower bound / upper bound). Solution files can be requested from the authors by email.

5.1 The Udine Benchmark Instances

5.1.1 Benchmarks from ITC2002

In Table 1 we list optimal solutions we generated for the four ITC2002 instances (not yet with the RS constraints) from the university of Udine.

For all except one instance, running times are quite short. Taking into account that no previous approach has produced optimal results for all four instances, this is remarkable and demonstrates the usefulness of our approach. Among all soft constraints, curriculum compactness appears to destroy the combinatorial structure of the timetabling problem the most. An impressive proof for this is given in Table 2 where these constraints are dropped.

In order to check the necessity of a commercial solver we tested the non commercial, open source solvers SCIP[1] (`scip.zib.de`) and CBC (`www.coin-or.org/Cbc/`) to solve our integer

instance	our obj	previous best obj	our status	previous status	CPU sec.
Test1	212	213	feasible, optimal	feasible	15.40
Test2	8	8	feasible, optimal	feasible	6.31
Test3	35	43	feasible, optimal	feasible	82.33
Test4	27	50	feasible, optimal	feassble	1607.30

Table 1: Optimal solutions for the ITC 2002 Udine problem instances without RS constraints. We list instance names; our objective function values (soft constraint penalties) and solution status as compared to the previously best known; and the CPU time needed for computations.

instance	obj	our status	CPU sec.
Test1	200	feasible, optimal	0.14
Test2	0	feasible, optimal	0.08
Test3	5	feasible, optimal	0.11
Test4	0	feasible, optimal	0.17

Table 2: Optimal solutions for the ITC 2002 Udine instances without the CC constraint

programs. These could not match the good running times of the commercial solver CPLEX, see Table 3. The use of a commercial solver (and thus, the possible lack of reproducibility of results on any machine) is, in fact, the reason why we did not submit our results to the ITC2007 competition.

5.1.2 Benchmarks from ITC2007

The second International Timetabling Competition, ITC2007, extended the definition of soft constraints. Seven instances were provided within the competition. In Table 4 we list our results separately for the two stages of the decomposition. The running time of the second stage is always bounded by 100 CPU seconds.

5.1.3 Extensions

In Section 4 we discussed several extensions for soft constraints as proposed in [7]. Tables 5 and 6 list our results for the ITC2002 and ITC2007 instances, when the problem definition is exemplarily extended by the *Maximum Dayload* and the *Lunch Break* constraints. We did not include the other extended soft constraints in this study.

5.2 Simulated Data from Technical University Berlin

As we have said, our original motivation was to keep hard constraints, if this is possible. At the Technical University of Berlin, room capacities are considered hard, and a number of features have to be provided by a room if requested by a course. This gives a much larger number of different room types. Since the used timetabling database is in an incomplete and inconsistent state, we developed a simulation tool which is able to create large problem

(a) CBC statistics after 400 sec.

instance	obj	lower bound	gap
Test1	231	200	13.41 %
Test2	40	0	100.00 %
Test3	-	16.98	infeasible
Test4	-	7.75	infeasible

(b) CBC statistics after 3600 sec.

instance	obj	lower bound	gap
Test1	216	200	8 %
Test2	23	0	100.00 %
Test3	146	17.748	87.84 %
Test4	254	7.75	96.94 %

(c) SCIP statistics after 400 sec.

instance	obj	lower bound	gap
Test1	218	212	2.75 %
Test2	14	0.0	100.00 %
Test3	-	23.79	infeasible
Test4	-	11.3	infeasible

(d) SCIP statistics after 3600 sec.

instance	obj	lower bound	gap
Test1	212	212	0 %
Test2	8	1.5	81.25 %
Test3	47	26.30	44.04 %
Test4	78	12.91	83.44 %

Table 3: Open source solver statistics for SCIP and CBC. We list objective function values, lower bounds, and optimality gaps. An entry "infeasible" means that no feasible solution was found within the given time frame.

(a) Objective function values after a maximal CPU sec. of 400 sec.

instance	obj stage 1	gap stage 1	obj stage 2	gap stage 2	obj	vio. hardc.
comp01	4	0 %	6	0 %	10	0
comp02	81	100.00 %	9	13.36 %	90	0
comp03	107	76.53 %	8	8.86 %	115	0
comp04	35	0 %	9	10.23 %	44	0
comp05	580	92.14 %	4	0 %	584	0
comp06	915	98.68 %	22	15.47 %	937	0
comp07	938	99.36 %	72	35.47 %	1010	0

(b) Objective function values after a maximal CPU sec. of 3600 sec.

instance	obj stage 1	gap stage 1	obj stage 2	gap stage 2	obj	vio. hardc.
comp01	4	0 %	6	0 %	10	0
comp02	57	96.64 %	5	5.57 %	62	0
comp03	92	68.32 %	14	15.12 %	106	0
comp04	35	0 %	8	9.20 %	43	0
comp05	426	80.66 %	4	0 %	430	0
comp06	67	82.09 %	28	19.43 %	95	0
comp07	6	0 %	67	33.84 %	73	0

(c) Objective function values without the CC constraints.

instance	obj stage 1	gap stage1	vio. hardc.	CPU sec.
comp01	4	0 %	0	0.1
comp02	0	0 %	0	0.85
comp03	0	0 %	0	0.3
comp04	0	0 %	0	0.6
comp05	15	0 %	0	1.54
comp06	0	0 %	0	1.79
comp07	0	0 %	0	1.60

Table 4: Solution statistics for the ITC2007 instances. We list the instance name; the objective function value and optimality gap, respectively, after the first and second stage of the decomposition; the final objective function value (soft constraint penalties); the number of violated hard constraints; and the CPU time.

instance	obj	lower bound	gap	our status	CPU sec.
Test1	217	215	0.97%	feasible	150
Test2	59	59	0%	feasible, optimal	26.23
Test3	127	127	0%	feasible, optimal	125
Test4	48	45.47	5.25%	feasible	> 3600

Table 5: Best solutions for the ITC2002 Udine problem instances, with extensions as discussed in Sections 4 and 5.1.3.

instance	obj	lower bound	gap	our status	CPU sec.
comp01	8	8	0%	feasible, optimal	11.42
comp02	417	35.71	92.12%	feasible	3600
comp03	202	59	70.07%	feasible	3600
comp04	28	28	0%	feasible, optimal	1183
comp05	418	120.73	71.12%	feasible	3600
comp06	96	11.08	88.45%	feasible	3600
comp07	407	3	99.26%	feasible	3600

Table 6: Solution statistics for the ITC2007 instances with extensions as discussed in Sections 4 and 5.1.3.

instances with near real-world character.

We present statistics of three representative instances of different sizes, cf. Table 7. The key data (not listed here) of the large instance is almost identical to that of Technical University of Berlin (which is a rather large university). We give running times for a preprocessing step necessary to generate only the actually needed Hall conditions (3), and for the two decomposition stages. These times are acceptable, even though for an interactive timetable design, some tuning would be necessary.

instance	courses	lectures	rooms	violations	preproc.	stage 1	stage 2
small	180	420	35	0	45 sec.	9 sec.	3 sec.
medium	950	2100	165	0	307 sec.	52 sec.	6 sec.
large	2100	4640	345	0	1235 sec.	5106 sec.	5 sec.

Table 7: Statistics and results for the instances from Technical University Berlin

6 Perspectives

Integer programming has been used in university course timetabling, in our view, predominantly because of its enormous modeling power. Going one step further, and exploiting the problem's structure, we are able to obtain solutions which respect all hard constraints. We are encouraged by our good results to further study the combinatorial structure hidden in soft constraints in order to exploit it in our model in a similarly successful manner.

From a practical point of view, one is interested in warm-starting computations from previous timetables in such a way, that small changes in the input result in small changes in the constructed timetable. This kind of *robustness* could be considered already in constructing the first timetable via the framework of robust optimization; however, this will require entirely new research efforts and is beyond the scope of this paper.

References

[1] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007. `http://opus.kobv.de/tuberlin/volltexte/2007/1611/`.

[2] E.K. Burke, J. Mareček, A.J. Parkes, and H. Rudová. On a clique-based integer programming formulation of vertex colouring with applications in course timetabling. Technical Report NOTTCS-TR-2007-10, The University of Nottingham, 2007. `arXiv:0710.3603v2`.

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

[4] M.W. Carter. A comprehensive course timetabling and student scheduling system at the university of Waterloo. In E. Burke and W. Erben, editors, *PATAT 2000: Proceedings of the 3th International Conference on the Practice and Theory of Automated Timetabling*, volume 2079 of *Lect. Notes Comp. Science*, pages 64–82, Berlin, 2001. Springer.

[5] 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.

[6] S. Daskalaki, T. Birbas, and E. Housos. An integer programming formulation for a case study in university timetabling. *European J. Oper. Res.*, 153:117–135, 2004.

[7] L. Di Gaspero, B. McCollum, and A. Schaerf. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Technical report, University of Udine, 2007.

[8] G. Lach and M.E. Lübbecke. Optimal university course timetables and the partial transversal polytope. Preprint 2007/45, TU Berlin, Institut für Mathematik, 2007.

[9] L. Lovász and M.D. Plummer. *Matching Theory.* North-Holland, Amsterdam, 1986.

[10] 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.

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

[12] K. Schimmelpfeng and S. Helber. Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, 29:783–803, 2007.