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

Gerald Lach* Marco E. Lübbecke*

February 27, 2008
Revised July 2, 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 not 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 on average compete well with or beat the previously best known solutions. Our algorithm is not an overall winner, but it is very robust in the sense that it deterministically gives satisfactory lower and upper bounds in reasonable computation time without particular tuning. For slightly larger instances from the literature our approach shows significant potential as it considerably beats previous benchmarks. We further present solutions of proven quality to a few much larger instances with more elaborate hard constraints.

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 obvious hard constraints are fulfilled: A teacher can only teach one course at a time, a lecture room cannot host two courses simultaneously, courses of the same curriculum must not be scheduled in parallel, etc. If this is impossible, the number of violations is to be minimized. Furthermore, several soft constraints should best possibly met; these typically give desired structural properties like coherent daily time slots for lectures of the same curriculum, etc. This problem, also known as university course timetabling, received much attention in the operations research literature, see the surveys [6, 18], not least due to the fact that practical data is available for benchmarking, in particular instances from the university of Udine [12, 13] used in ITC2002, the first International Timetabling

*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.

Competition. In ITC2007, the second issue of the competition (`www.cs.qub.ac.uk/itc2007`), more benchmarks from Udine were provided [11], 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 (which is also, but certainly not only due to the fact that they became rather easy for modern integer programming solvers), and give solutions which do not violate any hard constraint to the 2007 instances. Here it turns out that we are able to very well compete with, and often beat the strongest known, tailored solution methods which are based on heuristics. We furthermore provide solutions to instances derived from practical data 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., [3, 4, 5, 7, 8, 9, 17, 19]. However, instead of directly solving a natural formulation 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 consists 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 [14] we developed the theoretical background for our decomposition which considered hard constraints only. Here, 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 use in Berlin) 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., [3], 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 [15] 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 [14] on how to cope with this (and why in practice there are not too many of them).

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

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

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

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

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

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, that is, in principle it deterministically finds optimal solutions, provided one allows enough running time.

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 [11]. 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 [11].

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 (Berlin) 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. Hall's 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 [11] 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 currently 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

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

Clearly, if (17) is minimized over all feasible course/room assignments, the RS constraint is fulfilled best possibly 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 (and it happens) 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 $t \in \mathcal{T}$.

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

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

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

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

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

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

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.

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

4 Extensions

In [11] several more constraints are mentioned which are relevant in practice, but do not appear in the ITC2007 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 from Subsection 3.5.

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 [11], 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 $p_1, \dots, p_k$ 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 Study

We report on three different sets of experiments. In the first (Subsection 5.1), we deal with "the Udine instances," both from ITC2002 and ITC2007. The second set (Subsection 5.2) contains somewhat larger instances from a recent paper by Cesco, Di Gaspero, and Schaerf [10]. For both sets we consider both, the "basic" formulation [13] (without RS constraints), and the "extended" formulation [11] with all four types of soft constraints. The final (smallest) set (Subsection 5.3) contains much larger instances with only hard constraints. This last set reflects the timetabling situation at the Technical University of Berlin. All experiments were run on a 3.4GHz Linux PC with 1GB memory; unless specified otherwise, we solved integer programs with CPLEX 11.0.1. The reported optimality gaps were computed relative to the upper bound, i.e., as (upper bound − lower bound / upper bound).

The curriculum-based course timetabling web site `http://tabu.diegm.uniud.it/ctt/` is most helpful in making results comparable. First of all, they offer a solution validator which we used, of course, to validate our results (solution files can be requested from the authors by email). From the same web site one can download a program to benchmark machine speed. In our computations, one *CPU time unit* corresponds to the time allowed for one run in the ITC2007 competition: This should be around 400 seconds on a reasonable PC. For ITC2007, the program of every finalist was run 10 times, each time with a different random seed. Thus, it took 10 CPU time units to achieve their respective best solutions. Sometimes, the competition winner Tomáš Müller [16] did not achieve the overall best result for an instance. Since we also compare ourselves against these overall best solutions of all of the five finalists, we say that it took 5·10 CPU time units to obtain these solutions. When we compare ourselves to the best solutions by the university of Udine's Scheduling and Timetabling Group (SaTT) we assumed they used 40 CPU time units since they started 40 runs to obtain their best results. Since in contrast our approach is entirely deterministic, it is fair to allow ourselves a solution time equivalent to what is used in total in the respective runs of these various groups.

There are several (similar) tables, and if you are in a rush, the most important conclusions can be drawn from Tables 2, 5, 7, and 9.

5.1 The Udine Benchmark Instances

5.1.1 Benchmarks from ITC2002

In Table 1 we list for the first time proven optimal solutions for *all the four* ITC2002 instances, in particular `test4` was unsolved. These instances do not feature RS constraints.

	basic formulation [13]		without CC constraint	
instance	obj	CPU sec.	obj	CPU sec.
`test1`	212	15.40	200	0.14
`test2`	8	6.31	0	0.08
`test3`	35	82.33	5	0.11
`test4`	27	1607.30	0	0.17

Table 1: Optimal solutions values for the ITC2002 Udine problem instances (basic formulation [13]). We list instance names, our objective function values (soft constraint penalties), and the CPU time needed for computations. On the right the we see the results when the CC constraints are omitted.

For all except the last 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 (CC) appears to destroy the combinatorial structure of the timetabling problem the most. An impressive proof for this is given in Table 1 where these constraints are dropped.

The Role of the Solver It should be mentioned that the last years have seen great improvements in integer programming solvers, so one might suspect that our ability to solve `test1`–`4` is mainly due to this fact; however, with the several years old CPLEX9 we are able to produce optimal solutions to the first three instances within computation times comparable to those in Table 1, and a very good solution for `test4` (value 29) in about an hour. However, actually proving this quality is not possible with CPLEX9, since the lower bound does not improve at all (the zero-half cuts of later CPLEXes do help a lot in this respect).

In order to check the necessity of a commercial solver in the first place we tested the non commercial, open source solvers SCIP[1] (`scip.zib.de`) and CBC (`www.coin-or.org/Cbc/`) to solve our integer programs. These could not match the good running times of the commercial solver CPLEX. 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 by adding room stability (RS). Seven instances (`comp01`–`07`) were provided at the outset of the competition, seven more (`comp08`–`14`) followed closer to the deadline (and seven more after the deadline, but these are not yet available to us). Table 2 lists our results. As one can see we are always (except twice) better than the average run of the ITC2007 winner, and we are very competitive with the respective best results obtained by all the five finalists. Results obtained in Table 2 are with CPLEX' zero-half cuts turned on. In Table 3 we list statistics separately for the two stages of the decomposition for various overall time bounds. These results were obtained with CPLEX11 default parameter settings.

	basic formulation [13]		extended formulation [11]				
				ITC2007			
Instance	SaTT	us	winner ⊘	winner best	finalists best	SaTT	us
`comp01`	4	**4**	5.0	5	5	5	13
`comp02`	35	31	61.3	51	50	75	43
`comp03`	52	42	94.8	84	71	93	76
`comp04`	21	**18**	42.8	37	35	45	38
`comp05`	244	253	343.5	330	309	326	314
`comp06`	27	16	56.8	48	48	62	41
`comp07`	13	**3**	33.9	20	20	38	19
`comp08`	24	20	46.5	41	40	50	43
`comp09`	61	59	113.1	109	105	119	102
`comp10`	10	8	21.3	16	16	27	14
`comp11`	0	**0**	0.0	0	0	0	**0**
`comp12`	268	316	351.6	333	333	358	405
`comp13`	38	33	73.9	66	66	77	68
`comp14`	30	29	61.8	59	57	59	54
CPU time units	40	10	1	10	50	40	10

Table 2: We compare ourselves against the university of Udine's Scheduling and Timetabling Group (SaTT), against the objective of the ITC2007 winner, averaged over all his 10 runs, against his respective best run, and against the overall best run of all the five finalists.

(a) Overall time limit: 1 CPU time unit

	stage 1 (time limit: 300 sec.)						stage 2 (time limit: 80 sec.)				total
Instance	obj	LB	gap%	time	obj 1st	time 1st	obj	time	obj 1st	time 1st	obj
comp01	4	4.00	0.00	<5	4	<5	8	<1	8	<1	12
comp02	273	0.00	100.00	120	430	<50	16	20	147	2	239
comp03	191	0.00	100.00	261	468	<10	3	60	143	10	194
comp04	36	21.90	39.15	264	358	<5	8	40	144	10	44
comp05	956	91.83	90.39	290	1241	<90	9	<1	16	<1	965
comp06	346	7.00	97.98	280	541	<100	49	80	180	3	395
comp07	448	0.00	100.00	290	525	<190	68	80	225	3	525
comp08	39	29.20	25.11	190	344	<4	39	70	173	4	78
comp09	113	36.89	67.35	290	444	<2	2	80	160	2	115
comp10	194	2.00	98.97	200	425	<110	41	60	207	2	235
comp11	0	0.00	0.00	<1	0	<1	7	<1	7	<1	7
comp12	1119	28.08	97.49	290	1119	290	3	4	77	<1	1122
comp13	75	32.17	57.11	270	492	<3	23	80	161	2	98
comp14	110	39.50	71.79	290	449	<20	3	80	141	1	113

(b) Overall time limit: 10 CPU time units

	stage 1 (time limit: 3300 sec.)						stage 2 (time limit: 500 sec.)				total
Instance	obj	LB	gap%	time	obj 1st	time 1st	obj	time	obj 1st	time 1st	obj
comp01	4	4.00	0.00	<5	4	<5	8	<1	8	<1	12
comp02	93	8.00	91.40	∼ 3000	430	<120	0	208	138	2	93
comp03	84	0.00	100.00	3140	468	<40	2	300	132	10	86
comp04	35	27.43	21.61	2960	358	<5	5	330	145	10	41
comp05	463	24.30	95.26	2800	1241	<430	5	<1	13	<1	468
comp06	66	10.00	84.85	∼ 3000	541	<300	13	300	181	3	79
comp07	8	2.00	75.00	∼ 2000	525	<360	20	413	234	3	28
comp08	37	34.00	8.11	2990	344	<10	11	200	177	4	48
comp09	106	41.00	60.79	3280	444	<2	0	439	169	2	106
comp10	4	4.00	0.00	2385	425	<220	40	130	207	2	44
comp11	0	0.00	0.00	<1	0	<1	7	<1	7	<1	7
comp12	657	31.28	95.24	∼ 2500	1119	290	0	4	81	<1	657
comp13	61	38.60	36.72	∼ 1930	492	<3	6	300	155	3	67
comp14	51	41.00	18.66	∼ 1500	449	<20	3	284	146	1	54

(c) Overall time limit: 40 CPU time units

	stage 1 (time limit: 13000 sec.)						stage 2 (time limit: 2200 sec.)				total
Instance	obj	LB	gap%	time	obj 1st	time 1st	obj	time	obj 1st	time 1st	obj
comp01	4	4.00	0.00	<5	4	<5	8	<1	8	<1	12
comp02	45	10.33	77.04	∼ 11500	430	<120	1	2000	151	2	46
comp03	66	25.00	100.00	∼ 12500	468	<40	0	432	127	10	66
comp04	35	27.43	21.61	2960	358	<5	3	1300	51	10	38
comp05	365	107.97	70.43	12700	1241	<430	3	<1	38	<1	368
comp06	37	10.00	72.97	7526	541	<300	14	2000	187	3	51
comp07	6	6.00	0.00	10000	525	<360	19	2000	221	3	25
comp08	37	37.00	0.00	500	344	<10	8	1200	178	4	44
comp09	99	45.89	53.65	12800	444	<2	0	500	165	2	99
comp10	4	4.00	0.00	2385	425	<220	12	2000	207	2	16
comp11	0	0.00	0.00	<1	0	<1	7	<1	7	<1	7
comp12	546	52.70	90.34	11000	1119	290	1	4	79	<1	548
comp13	61	40.81	33.72	∼ 1930	492	<3	5	800	155	3	66
comp14	51	45.94	9.92	∼ 1500	449	<20	2	900	146	1	53

Table 3: Computation times for ITC2007 (extended formulation [11]) listed separately for the two decomposition stages (in seconds), for time limits of a total of 1, 10, and 40 CPU time units in the different subtables (a)–(c) (detailed limits on the two stages are given in the respective headings). We report (in that order) the instance name, the objective function value (plus lower bound and optimality gap), and the time to reach that solution. As a reference we also give the objective value of and the time to reach the first feasible integer solution. Information on the second stage is similar. Computation times listed in this table are rather coarsely reported and only serve as an indicator. CPLEX11 is used with default parameter settings.

5.1.3 Lower Bounds

Meta heuristics are powerful to produce solutions to quite large timetabling instances. However, assessing the quality of these solutions is much harder. Recently, Burke et al. [2] proposed a branch-and-cut algorithm to obtain lower bounds for the ITC 2007 instances. We note that the time to solve our linear programming (LP) relaxation is much smaller since our formulation contains much less variables, cf. Table 4. Further, the program presented in [2] is not yet suited to produce feasible integer solutions; this is why Burke et al. resorted to heuristics for this. We list the lower bounds obtained by our approach in Table 5.

	Burke et al. [2]		first stage IP			second stage IP		
Instance	#vars	#cons	#vars	#cons	#nonz	#vars	#cons	#nonz
comp01	6516	5500	4489	3843	16206	346	205	685
comp02	30128	26703	7872	7729	41117	3894	1525	9949
comp03	26941	24563	8184	8099	41161	3419	1379	8749
comp04	33698	30525	10637	10242	47212	4171	1600	10755
comp05	16259	12129	9688	9823	56111	457	278	1034
comp06	44168	40113	14810	14421	68821	5472	2056	13923
comp07	60745	55895	17220	16959	80669	7845	2709	19904
comp08	35735	32397	11426	11123	49723	5080	1785	13203
comp09	32391	29024	9113	8833	42276	4454	1651	11506
comp10	45996	42279	12428	12123	59549	6252	2285	15814
comp11	8733	6672	4385	4153	22122	243	159	617
comp12	27652	22117	10735	10705	64605	2208	1115	5307
comp13	35691	32353	11781	11412	51793	4706	1679	12200
comp14	33384	30057	9878	9678	48207	4140	1686	10407
DDS1			16209	15305	76215	14998	5119	39255
DDS2			3287	2201	20688	1669	952	3741
DDS3			2274	1291	10878	2048	1038	5336
DDS4			102384	99502	459895	19327	5951	50827
DDS5			19429	17440	82261	9266	3173	24619
DDS6			12845	12584	60641	5090	2006	12730
DDS7			5622	4701	37800	2023	713	4044

Table 4: Sizes of the formulation proposed by Burke et al. [2], and of our integer programs after presolve: Numbers of variables, constrains, and non zero elements are listed. Our (time consuming) first stage IP is about a factor of three (in each dimension) smaller than the formulation proposed in [2].

5.1.4 Extensions

In Section 4 we discussed several extensions for soft constraints as proposed in [11]. Table 6 lists our results for the ITC2002 and the first seven 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.

	root relaxation (sec.)		LB (root)		LB (after 30 min.)	
Instance	[2]	us	us	us w/ cuts	[2]	us
comp01	3.58	0.09	4	4	5	4
comp02	54.90	0.68	0	0	6	8
comp03	49.97	0.59	0	1	43	23
comp04	41.00	0.12	0	11.5	2	26.27
comp05	84.64	1.97	17	92.45	183	100.9
comp06	73.17	0.88	6	7	6	7
comp07	192.35	1.47	0	0	0	0
comp08	43.25	0.61	0	1.16	2	33.2
comp09	48.23	0.51	0	18.2	0	39.84
comp10	105.03	1.00	0	2	0	3.91
comp11	7.69	0.13	0	0	0	0
comp12	134.76	2.92	3	30.25	5	31.29
comp13	37.34	0.67	0	20	0	37
comp14	55.86	0.72	0	39.5	0	41

Table 5: Lower bounds (LBs) obtained by Burke et al. [2] and with our approach (first stage) for the extended formulation [11]. On the left one can see the computation times to solve the LP relaxation, then the LBs in the root node with our plain formulation, and after adding CPLEX' zero-half cuts; finally, the LBs after half an hour computation time. Numbers are taken from a first draft of [2]; updated results are not available to us.

instance	obj	lower bound	gap	status	CPU sec.
test1	217	215	0.97%	feasible	150
test2	**59**	59	0.00%	optimal	26.23
test3	**127**	127	0.00%	optimal	125
test4	48	45.47	5.25%	feasible	3600
comp01	**8**	8	0.00%	optimal	11.42
comp02	417	35.71	92.12%	feasible	3600
comp03	202	59	70.07%	feasible	3600
comp04	**28**	28	0.00%	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: Best solutions for the ITC2002 and the first seven ITC2007 problem instances, with extensions as discussed in Sections 4 and 5.1.4. Bold face marks optimal solutions.

5.2 Instances with More Courses

A hint on the potential of our approach when applied to larger instances is given on data recently introduced by Cesco, Di Gaspero, and Schaerf [10]. Some of them have a (slightly) larger number of courses (`DDS1` and `DDS4`), and our integer program performs significantly better than what was previously known, see Table 7.

	basic formulation [13]		extended formulation [11]	
Instance	SaTT	us	SaTT	us
`DDS1`	238	**39**	1024	132
`DDS2`	0	**0**	0	**0**
`DDS3`	0	**0**	0	**0**
`DDS4`	233	19	261	68
`DDS5`	0	**0**	0	**0**
`DDS6`	5	**0**	11	4
`DDS7`	0	**0**	0	**0**
CPU time units	40	10	40	10

Table 7: Slightly larger instances taken from [10]; our approach compared to the university of Udine's Scheduling and Timetabling Group (SaTT); bold face indicates optimal solutions. Zero-half cuts are turned on in these computations.

5.3 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 (internet access, PC/beamer, blackboard, location, etc.). This gives a much larger number of different room types, but in general fewer eligible rooms per course. All other soft constraints presented here are not respected, as they are not relevant for this university. Since the used timetabling database is in an incomplete and inconsistent state, we decided to develop a simulation tool which is able to create large problem instances with near real-world character.

We present statistics of three representative instances of different sizes, cf. Table 9. 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.

(a) Overall time limit: 1 CPU time unit

	stage 1 (time limit: 300 sec.)						stage 2 (time limit: 80 sec.)				total
Instance	obj	LB	gap%	time	obj 1st	time 1st	obj	time	obj 1st	time 1st	obj
DDS1	147	42.99	70.75	<240	237	<200	144	60	499	<5	291
DDS2	0	0.00	0.00	<1	60	<1	0	<5	47	<5	0
DDS3	0	0.00	0.00	<1	22	<1	0	60	78	<5	0
DDS4	675	15.00	97.78	300	1067	<300	376	40	484	<5	1051
DDS5	0	0.00	0.00	22	147	<2	42	70	298	<5	42
DDS6	159	0.00	0.00	280	460	<100	27	80	171	<5	186
DDS7	0	0.00	0.00	52	127	<3	4	80	110	<5	4

(b) Overall time limit: 10 CPU time units

	stage 1 (time limit: 3300 sec.)						stage 2 (time limit: 500 sec.)				total
Instance	obj	LB	gap%	time	obj 1st	time 1st	obj	time	obj 1st	time 1st	obj
DDS1	48	48.00	0.00	2000	237	<200	55	800	499	<5	103
DDS2	0	0.00	0.00	<1	60	<1	0	< 5	47	<5	0
DDS3	0	0.00	0.00	<1	22	<1	0	60	78	<5	0
DDS4	17	15.00	11.76	700	1067	<300	95	500	484	<5	112
DDS5	0	0.00	0.00	22	147	<2	10	800	298	<5	10
DDS6	4	0.00	100.00	1000	460	<100	5	500	171	<5	9
DDS7	0	0.00	0.00	52	127	<3	0	309	110	<5	0

(c) Overall time limit: 40 CPU time units

	stage 1 (time limit: 13000 sec.)						stage 2 (time limit: 2200 sec.)				total
Instance	obj	LB	gap%	time	obj 1st	time 1st	obj	time	obj 1st	time 1st	obj
DDS1	48	48.00	0.00	2000	237	<200	35	1800	499	<5	83
DDS2	0	0.00	0.00	<1	60	<1	0	<5	47	<5	0
DDS3	0	0.00	0.00	<1	22	<1	0	60	78	<5	0
DDS4	17	15.00	11.76	700	106	<300	75	2000	484	<5	92
DDS5	0	0.00	0.00	22	147	<2	10	700	298	<5	10
DDS6	0	0.00	0.00	3000	460	<100	3	2000	171	<5	3
DDS7	0	0.00	0.00	52	127	<3	0	309	110	<5	0

Table 8: Solution statistics reported separately for the two decomposition stages for the DDS1–7 instances, in the same way as we did in Table 3. Default parameter settings are used.

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 9: Statistics and results for the simulated instances according to Technical University of Berlin's course database

6 Perspectives

Integer programming has been used in university course timetabling, in our view, predominantly because of its enormous modeling power. Only recently, researchers started to exploit the problem's structure, as we did in this paper. 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.

Our approach was meant to solve instances from Berlin's technical university where all constraints are considered hard, see [14]. However, feedback on that approach motivated us to evaluate its suitability for incorporating soft constraints, or to obtain lower bounds. We believe it is fair to say that our algorithm certainly is not best in all possible situations, however, it competes quite well for several different purposes, as is demonstrated by our computational study. This is even more true since we do not use any particular tuning to the instances or situation (lower or upper bound). In that sense, the proposed approach may serve as a very *robust* starting point for more ambitious goals in timetabling.

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 other 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] Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. A branch-and-cut procedure for the udine course timetabling problem. In E.K. Burke and M. Gendreau, editors, *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008*, Montréal, CA, 2008.

[3] 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`.

[4] E.K. Burke, J. Mareček, A.J. Parkes, and H. Rudová. Penalising patterns in timetables: Strengthened integer programming formulations. In J. Kalcsics and S. Nickel, editors, *Operations Research Proceedings 2007*, pages 409–414, Berlin, 2008. Springer.

[5] E.K. Burke, J. Mareček, A.J. Parkes, and H. Rudová. Uses and abuses of MIP in course timetabling. Poster at the Workshop on Mixed Integer Programming, MIP2007, Montréal, 2008. Available online at `http://cs.nott.ac.uk/~jxm/timetabling/mip2007-poster.pdf`.

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

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

[8] 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.

[9] 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.

[10] F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, and results. In E.K. Burke and M. Gendreau, editors, *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008*, Montréal, CA, 2008.

[11] 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.

[12] L. Di Gaspero and A. Schaerf. Multi neighborhood local search with application to the course timetabling problem. In E. Burke and P. De Causmaecker, editors, *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2002*, volume 2740 of *Lect. Notes Comp. Science*, pages 262–275, Berlin, 2003. Springer.

[13] L. Gaspero and A. Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *J. Math. Modelling and Algorithms*, 5:65–89, 2006.

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

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

[16] T. Müller. ITC2007 solver description: A hybrid approach. In *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008*, Montréal, CA, 2008.

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

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

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