Ann Oper Res (2012) 194:439–454
DOI 10.1007/s10479-010-0708-z

An IP-based heuristic for the post enrolment course timetabling problem of the ITC2007

J.J.J. van den Broek · C.A.J. Hurkens

Published online: 12 February 2010

Abstract Track 2 of the international timetabling competition 2007 was a post enrolment course timetabling problem. A set of events has to be assigned to a timeslot and to a room such that all students are able to attend their requested events while not violating the hard constraints. There are also soft constraints that make the timetable "nicer".

We present a deterministic heuristic that assigns events to timeslots based on an LP-solution constructed with column generation. We get an integer solution by fixing columns one at a time. This heuristic finds a solution that obeys all the hard constraint for 23 of the 24 instances of the competition. The generated solution is improved by selecting a set of events that are reassigned by solving an integer program. This IP minimizes the number of soft constraint violations under the restriction that no hard constraints are violated. Comparing the results of our heuristic with the results of the five finalists of the competition, shows that our approach is competitive.

Keywords Course timetabling · ITC2007 · Column generation · Integer programming

1 Introduction

One of the major difficulties in the field of university timetabling is that the proposed algorithms are mostly tested on data of the university of the researchers. Above this almost all universities have different timetabling problems, mainly different types of constraints. This makes a fair comparison between different algorithms very difficult. Therefore, the second international timetabling competition (ITC2, International Timetabling Competition 2007) has been organized. The competition contained three tracks of which the second was the post-enrolment course timetabling problem described in this paper.

J.J.J. van den Broek (✉) · C.A.J. Hurkens
Department of Mathematics and Computer Science, Eindhoven University of Technology,
Den Dolech 2, 5600 MB Eindhoven, The Netherlands
e-mail: j.j.j.v.d.broek@tue.nl

C.A.J. Hurkens
e-mail: wscor@win.tue.nl

Course timetabling is the weekly scheduling of a set of lectures for a set of university courses within a set of rooms and timeslots, minimizing the overlaps of lectures of courses having common students, Schaerf (1999). In a post enrolment course timetabling problem students select the lectures they wish to attend. The timetable is constructed according to these choices and no overlap is allowed. The post enrolment course timetabling problem in ITC2 was also used in ITC1, but in ITC2 two extra hard constraints are introduced to make it more difficult to find a feasible timetable.

Carter and Laporte (1998) provide a survey on practical course timetabling problems. Especially the last decades local search techniques became popular for solving course timetabling problems, see for example Kostuch (2004) and Rossi-Doria et al. (2002). Also some of the finalists of track 2 of the ITC2 applied a local search technique. Atsuta et al. (2008) use a general CSP solver. This solver is an hybrid algorithm of tabu search and iterated local search. Cambazard et al. (2008) use several techniques, in particular simulated annealing, constraint programming and hybrids of those in a large neighborhood search scheme. With their simulated annealing augmented with a large neighborhood search move at the low temperatures they won the competition. A stochastic local search approach consisting of several heuristic modules in different phases of the solution process is applied by Chiarandini et al. (2008). All modules are tuned with the automated algorithm configuration procedure ParamILS (Hutter et al. 2007). They were the only finalist that found feasible solutions for all instances. Müller (2008) developed a constraint-based framework containing a series of local search algorithms that operate over feasible, not necessarily complete, solutions. His solver won the tracks 1 and 3 of the ITC2007 and ended number 5 in track 2. The fifth and last finalist of track 2 was Mayer et al. (2008). They developed a heuristic based on ant colony optimization.

Our main contribution is a totally different solution approach for the problem of track 2. We use integer programming techniques and introduce a new heuristic using column generation. A compact integer programming formulation of the problem was formulated. Solving this ILP with CPLEX was impossible within the allowed computation time of the competition. For some instances CPLEX was even not able to solve the LP-relaxation within a few minutes. Therefore we decided to develop heuristics.

The next section provides a description of the problem (Lewis et al. 2007) and introduces our notation. In Sect. 3 a construction heuristic is described that assigns events to timeslots based on an LP-solution that we get by applying column generation on an extended ILP formulation of the problem. After construction of a solution we try to improve the solution by solving small subproblems to optimality with a compact ILP formulation. Our improvement procedure is presented in Sect. 4. Section 5 gives our results and compares them with the results of the finalists.

2 Problem description

Given is a set E of events that have to be assigned to a timeslot from the set T of timeslots ($T = \{0, \ldots, 44\}$) and to a room from the set R of rooms. Every room $r \in R$ has a set of features F_r and a specific seating capacity C_r. Above this we have a set S of students that all have a set E_s of events that they are attending. Every event e has a set T_e of timeslots to which it can be assigned, a set F_e of features that it requires and N_e students attending the event. An event e can be assigned to room r if room r satisfies all features that event e requires ($F_e \subseteq F_r$) and the seating capacity of room r is sufficient for the number of students participating ($N_e \leq C_r$). We define R_e as the set of rooms to which event e can be assigned.

Also precedence constraints are given that state that certain events should be scheduled earlier in the week than other events. Precedences between events are modeled with the parameter p_{ef}. It has value one if e is a predecessor of f.

In summary: The input consists of:

- set T of 45 timeslots $\rightarrow T = \{0, \ldots, 44\}$.
- set R of rooms.
- set E of events.
  - $\forall e, f \in E : p_{ef} = 1$ if event e has to be scheduled before event f.
  - $\forall e \in E : N_e =$ number of students attending event e.
  - $\forall e \in E :$ set T_e of timeslots to which event e can be assigned.
  - $\forall e \in E :$ set R_e of rooms to which event e can be assigned.
- set S of students.
  - $\forall s \in S :$ set E_s of events that student s is attending.

Two events are *colliding* if they can not be assigned to the same timeslot. This arises if they have a student in common, if both have only one possible room which is the same or if there is a precedence relation between the two events. We define c_e as the number of events colliding with event e.

Above the fact that a timetable has to be created such that all students can attend the events they request, a timetable should be 'nice'. Therefore, the constraints in the timetabling problem can be divided into soft- and hard constraints. The hard constraints need to be satisfied in order for the timetable to be feasible. Soft constraints make the timetable nicer for the people who are supposed to use it and have to be satisfied as much as possible.

The five hard constraints that have to be satisfied to produce a feasible timetable are:

1. No student can attend more than one event at a time.
2. An event e can only be assigned to a room $r \in R_e$.
3. At most one event is assigned to each room in any time slot.
4. An event e can only be assigned to a time slot $t \in T_e$.
5. Events have to be scheduled in the prescribed order during the week.

The soft constraints that have to be satisfied are:

- Events should not be assigned to the last timeslots of a day (that is: $t = 8, 17, 26, 35$ and 44).
- Students should not have to attend three or more events in successive timeslots on the same day.
- Students should not be required to attend only one event a day.

Because feasibility could be difficult the organizers made the distinction between a *valid timetable* and a *feasible timetable*. For both timetables hard constraint violations are not allowed. For a valid timetable it is allowed that events are left out of the timetable. A feasible timetable is one in which there are no hard constraint violations and all events are assigned. All solutions have to be valid. The quality of valid solutions is evaluated with two measures:

1. Distance to feasibility (dtf)
2. Total number of violated soft constraints.

The *distance to feasibility* is equal to the number of students that require an unplaced event. A feasible timetable always has a distance to feasibility of zero. The valid solution with the lowest distance to feasibility is the better solution. If two valid solutions have the same distance to feasibility, then the solution with the minimum number of violated soft constraints is preferred.

3 Heuristic based on LP-solution

It is possible to formulate the problem as a compact integer program, including the soft constraints. Such an IP formulation will be presented in Sect. 4. In the current section the focus is on constructing a feasible solution. Even solving the compact IP formulation without the soft constraints is not possible within the allowed computation time of the ITC. Even worse, CPLEX did not start branching within this allowed computation time. Therefore, we formulated the problem without the soft constraints as an extended integer programming formulation. We solve the LP-relaxation of this extended ILP using column generation. After solving the LP-relaxation we assign a set of events to a certain timeslot and solve the LP-relaxation again. By solving the LP-relaxation every iteration, we guarantee that the events are spread over the week.

An introduction to column generation is provided by Desrosiers and Lübbecke (2005) and Lübbecke and Desrosiers (2005). They also give an overview of recent papers on this topic. Numerous applications in which column generation is used are described in literature. Desaulniers et al. (2001) apply column generation for vehicle routing problems. Crew scheduling is another hot topic for which column generation is used a lot, see for example Desaulniers et al. (2001) and Borndörfer et al. (2003). Much more examples of applications can be found in Desrosiers and Lübbecke (2005). A far as we know nobody ever applied a column generation approach on a course timetabling problem.

Our main goal of the heuristic described in this section is the construction of a good feasible solution. Therefore, our focus is on assigning all events to a room and timeslot without violating one of the hard constraints. In the first subsection the LP-formulation is introduced. The column generation procedure to solve this LP is described in the second subsection. How we generate new columns is presented in Sect. 3.3. Section 3.4 covers the heuristic that constructs an integer feasible solution using the feasible solution of the LP.

3.1 The LP formulation

The LP-formulation described in this subsection is a formulation of the timetabling problem without the soft constraints. We first introduce K as the set of *slot-schedules*. A *slot-schedule* $k \in K$ is characterized by a timeslot t_k, a set E_k of events and a matching $\rho_k : E_k \to R$. The matching ρ_k describes a feasible room assignment for all events in the slot-schedule. A slot-schedule k is feasible if:

- all $e \in E_k$ are not colliding.
- $\forall e \in E_k : t_k \in T_e$.
- $\forall e \in E_k : \rho_k(e) \in R_e \ \& \ \forall r \in R : |\{e|\rho_k(e) = r\}| \leq 1$.

For all slot-schedules $k \in K$ and all events $e \in E$ we introduce the parameter a_{ek}, that is one if $e \in E_k$ and zero otherwise.

The LP-formulation has three different types of decision variables. These are:

- $\forall k \in K : x_k$ is the number of times slot-schedule k is selected.
- $\forall e \in E : y_e = 0$ if event e is assigned to a timeslot, strictly positive otherwise.
- $\forall e, f \in E, p_{ef} = 1 : z_{ef} \in \mathbb{Z}_+$ is the violation of the precedence constraint between e and f.

Note that a feasible solution has y_e and z_{ef} equal to zero. The constraints together with the objective function ensure that decision variables x and y are not set to higher than one.

Now the decision variables and parameters have been introduced, we are able to present the LP:

$$\min \sum_{e \in E} y_e + \sum_{e,f \in E | p_{ef}=1} z_{ef}$$

$$y_e + \sum_{k \in K} a_{ek} x_k \geq 1 \quad \forall e \in E, \tag{1}$$

$$\sum_{k \in K | t_k = t} x_k \leq 1 \quad \forall t \in T, \tag{2}$$

$$z_{ef} + \sum_{k \in K} t_k (a_{fk} - a_{ek}) x_k \geq 1 \quad \forall e, f \in E | p_{ef} = 1, \tag{3}$$

$$x_k \geq 0 \quad \forall k \in K, \tag{4}$$

$$y_e \geq 0 \quad \forall e \in E, \tag{5}$$

$$z_{ef} \geq 0 \quad \forall e, f \in E | p_{ef} = 1 \tag{6}$$

Constraint (1) together with the first term of the objective function ensure that all events are assigned at least once. If the decision variable y_e is set to zero this implies that a slot-schedule containing event e is used. For every timeslot at most one slot-schedule can be selected, which is enforced by constraint (2). A precedence relation between two events is taken care of by constraint (3) together with the second part of the objective function.

3.2 Solving the relaxed LP-formulation with column generation

There are instances with a lot of precedence constraints which caused a too large computation time for solving the LP. Therefore, we choose to relax the LP-formulation by leaving out the precedence relations (3). The precedence relations are taken care of in the procedure described in Sect. 3.4. Our master problem (MP) is the LP-formulation without the precedence constraints.

We could consider the complete set of possible slot-schedules, but we cannot handle such a big LP and it would already cost too much computation time to generate them. Therefore we try to solve MP using column generation. The restricted master problem (RMP) is MP with a restricted set of possible slot-schedules K^r. The main idea of the column generation approach is to generate only the slot-schedules that with high probability are in the basis of an optimal feasible solution of MP.

Solving RMP gives shadowprices for all constraints. The *shadowprice* of a particular constraint of an LP is the value of the optimal dual solution variable associated with that constraint. It also represents the change in the value of the objective function per unit increase or decrease of the right-hand-side value of that constraint. So shadowprices give us information that we use to generate good feasible slot-schedules. We denote the shadowprices of constraint (1) by α_e and of constraint (2) by β_t.

To start with a column generation procedure enough variables are needed to find an initial basis. The y_e variables are in the initial formulation and for each timeslots in T we create a slot-schedule k with $E_k = \emptyset$.

After solving RMP we use the shadowprices to generate ("to price") new columns. These columns are generated by the *column generator* that we describe in detail in Sect. 3.3. The column generator generates feasible slot-schedules for a certain timeslot t that can be added

to the set K^r. The input of the column generator consists of a set E^{cg} of events that can be scheduled on timeslot t. Then the reduced cost c^k of slot-schedule k, generated by the column generator is:

$$c^k = \sum_{e \in E_k} \alpha_e + \beta_t$$

We proceed generating new columns until for all timeslots no new slot-schedules are found or until a certain amount of columns has been generated. Therefore, the column generation procedure does not always generate an optimal solution of MP.

The column generator generates a set K^{cg} of feasible slot-schedules. We are only interested in slot-schedules that are with high probability in an optimal MP-solution. Therefore a slot-schedule $k \in K^{cg}$ is only added to K^r if the reduced costs c^k of the column are larger than zero and larger than a percentage of the average reduced costs $\overline{c}$ of the most recently added slot-schedules.

We now present our column generation procedure in more detail:

1. Initialize K^r, set $t = 0$ and $factor = 0.75$.
2. Solve RMP $\rightarrow \alpha_e$, β_t and obj^{rmp}.
3. If $obj^{rmp} \leq 0$, then quit.
4. Generate set K^{cg} of feasible slot-schedules for timeslot t.
5. Add column $k \in K^{cg}$ to K^r if $c^k > factor * \overline{c}$. Add at most 10 columns.
6. $t = t + 1$
7. If $|K^r| > 2(|E| + |T|)$ or $\forall t \in T$ no column has been added, then quit.
8. Go to step 2.

In step 3 we quit if the objective value of RMP is zero, because then we found an optimal solution of MP. Step 4 generates feasible slot-schedules for a certain timeslot. This procedure generates 120 feasible slot-schedules as is described in the next subsection. To take care of enough varying columns, only the best 10 columns are added. To prohibit that too much computation time is used for generating columns the column generation procedure stops if we have a number of columns that is twice the number of rows. In step 7 the column generation procedure is terminated if this number of columns in RMP is reached or when there is no promising column found for any timeslot.

3.3 The column generator

The column generator generates a set K^{cg} of feasible slot-schedules for a certain timeslot t. The input of the column generator is a set $E^{cg} \subseteq E$ of events with for each event $e \in E^{cg}$ a corresponding shadowprice α_e. The goal is to generate feasible slot-schedules with a large sum of the shadowprices of the assigned events.

It is possible to determine the slot-schedule with the maximum sum of the shadowprices. This can be done by solving an IP that uses the decision variable $y'_e = 1 - y_e$ that is one if event e is assigned to a room, else it is zero. Also the following decision variable is defined $\forall r \in R_e$:

$$y_{er} := \begin{cases} 1 & \text{if event } e \text{ is assigned to room } r \\ 0 & \text{otherwise} \end{cases}$$

The integer programming problem is:

$$\max \sum_{e \in E^{cg}} \alpha_e y'_e$$

$$\sum_{r \in R_e} y_{er} = y'_e \qquad \forall e \in E^{cg}, \tag{7}$$

$$\sum_{e \in E^{cg} | e \in E_s} y'_e \leq 1 \qquad \forall s \in S, \tag{8}$$

$$\sum_{e \in E^{cg} | r \in R_e} y_{er} \leq 1 \qquad \forall r \in R, \tag{9}$$

$$y'_e + y'_f \leq 1 \qquad \forall e, f \in E^{cg} | p_{ef} = 1, \tag{10}$$

$$y_{er} \in \{0, 1\} \quad \forall e \in E^{cg}, \forall r \in R_e, \tag{11}$$

$$y'_e \in \{0, 1\} \quad \forall e \in E^{cg} \tag{12}$$

Constraint (7) takes care that an event is assigned to at most one room or it is not assigned. Students with more than one event assigned are not allowed, this is fulfilled by constraint (8). Constraint (9) imposes that only one event is assigned to each room. If there is a precedence relation between two events they can not be assigned both, this is enforced by constraint (10).

Solving this IP to optimality and adding the generated column to RMP would guarantee that we find an optimal solution of MP (Lübbecke and Desrosiers 2005). But unfortunately solving the IP takes too much computation time. Therefore, we developed a procedure that generates quickly a set of feasible slot-schedules by a greedy assignment procedure. This procedure is the central topic of the rest of this Section.

The procedure starts with sorting all $e \in E^{cg}$ according to the following criteria:

1. Decreasing order of α_e.
2. Decreasing order of c_e.
3. Decreasing order of N_e.
4. Increasing order of $|R_e|$.
5. Increasing order of $|T_e|$.

We denote the sorted set of events with E_s^{cg}. The procedure generates 120 different columns by selecting 120 times a set of three not colliding events on top of the sorted list for which a room assignment is possible. After those three events are assigned we try to greedily fill up the slot-schedule. The full procedure to generate columns will now be presented in pseudo code:

```
1:  K^cg = ∅.
2:  for (first 20 events e_1 of E_s^cg) do
3:    for (first 3 events e_2 of E_s^cg after e_1 not colliding with e_1) do
4:      Assign e_1 and e_2 to rooms.
5:      for (first 2 events e_3 of E_s^cg after e_2 not colliding with e_1 and e_2 and with possible
        room assignment) do
6:        Assign e_3 to a room.
7:        repeat
8:          e is next event of E_s^cg.
9:          if (e does not collide with an already assigned event) and (e can be assigned
            to a room) then
10:           Assign e to a room.
11:         end if
12:       until (There is no event left in E_s^cg)
13:       Add generated slot-schedule to K^cg.
```

14: **end for**
15: **end for**
16: **end for**

Every time an event e is a candidate for adding it to a slot-schedule it has to be verified whether there is a room available for e given the current room assignment of the already assigned events. Assigning events to rooms is a bipartite matching with the set of rooms and set of events as the vertices of the graph. Two vertices are connected if an event can be assigned to a room. Given the already assigned events to rooms it costs at most one augmenting path step to determine whether e can be added to the given matching.

3.4 Fix generated columns

Solving MP as described in the last subsections provides a solution for MP which is not necessarily optimal. The solution is probably even not feasible for the LP-formulation, because some of the precedence relations are probably violated. The generated solution consists of non-integer x-variables. To get a feasible integer solution we developed a heuristic that fixes one column at a time, which is the same as setting one x-variable to one. This heuristic is described in this section.

During the column generation procedure a lot of feasible slot-schedules are generated. Of those generated slot-schedules we fix the generated slot-schedule k with the highest value of $obj_k = c_k - penalties$ with $c_k = \sum_{e \in E} a_{ek} c_e$. A penalty is subtracted if one or more events are left with hardly any possible timeslots after selecting the slot-schedule. Also we subtract a penalty if fixing a column would violate a lot of three in a row constraints.

After fixing a column we delete a lot of slot-schedules that have been generated in the last iterations. Of course columns containing events that were in the fixed column and columns with the same timeslot are deleted. Also slot-schedules that are not feasible anymore, because of violated precedence constraints, are deleted. Next to those we delete "bad" columns. An example of such a column: If an event e has to be assigned earlier in the week then f, we delete a slot-schedule with timeslot on the first day if it contains f. Deletion of all these columns gives an enormous speed up in solving MP.

Before we present the heuristic, we define the set E_c of events that have not been assigned and the set T_c of timeslots that do not have a fixed slot-schedule. The heuristic to construct a valid solution goes as follows:

1. Initialize $T_c = T \backslash \{8, 17, 26, 35, 44\}$ and $E_c = E$.
2. Apply column generation procedure $\rightarrow K^r$.
3. Fix slot-schedule $k' = \max_{k \in K^r} obj_k$.
4. $T_c = T_c \backslash t_{k'}$, $E_c = E_c \backslash E_{k'}$ and delete columns.
5. If $|T_c| > 5$ and $|E_c| > 0$, then go to step 2.
6. If $|E_c| > 0$, then solve the compact IP presented in Sect. 4 to assign all $e \in E_c$ to a timeslot $t \in T_c \cup \{8, 17, 26, 35, 44\}$.

To take into account the soft constraint of no events on the last timeslot of the day, these timeslots are not added to the set T_c. In step 6 we solve the compact formulation of the problem with the events that are left. This IP is introduced in Sect. 4. Because CPLEX is able to find a very good feasible solution quickly for the last 10 timeslots we quit in step 5 when $|T_c| = 5$.

4 Improvement heuristic by integer programming

Section 3 introduced a heuristic that constructs a valid solution. The heuristic focuses on assigning the events to timeslots and does hardly take into account the soft constraints. Constructing this valid solution costs only a small part of the computation time that was allowed within the competition. Therefore, we also developed a procedure that improves the solution constructed by the LP-based heuristic. The first subsection presents an integer programming formulation of the timetabling problem including all soft constraints. The introduced ILP is used to improve the solution found by the LP-based heuristic. Section 4.2 presents our improvement heuristic.

4.1 The ILP-formulation of the problem

We define the set D of days and the set T' as the set containing the first 7 timeslots of all days. All other parameters have been introduced in Sect. 2. Left to define are the decision variables:

$$z_{ert} := \begin{cases} 1 & \text{if event } e \text{ is assigned to room } r \text{ on timeslot } t \\ 0 & \text{otherwise} \end{cases}$$

$$y_e := \begin{cases} 1 & \text{if event } e \text{ is not assigned} \\ 0 & \text{otherwise} \end{cases}$$

$$u_e := \begin{cases} 1 & \text{if event } e \text{ is assigned to a last timeslot of the day} \\ 0 & \text{otherwise} \end{cases}$$

$$v_{st} := \begin{cases} 1 & \text{if student } s \text{ is assigned to an event on timeslots } t, t+1 \text{ and } t+2. \\ 0 & \text{otherwise} \end{cases}$$

$$w_{sd} := \begin{cases} 1 & \text{if student } s \text{ is assigned to exactly one event on day } d \\ 0 & \text{otherwise} \end{cases}$$

$$w'_{sd} := \begin{cases} 1 & \text{if student } s \text{ is assigned to at least one event on day } d \\ 0 & \text{otherwise} \end{cases}$$

Events can not be assigned to all possible timeslots and rooms. Therefore, z_{ert} is only defined for the triples (e, r, t) if $t \in T_e$ and $r \in R_e$. This already ensures hard constraints two and four.

The objective function consists of two parts. The first part of the objective function takes care that the number of students that have a requested event not assigned is minimized. The second part consists of three terms that account for the number of soft constraint violations. Finding a solution with a distance to feasibility of zero is the most important goal. Therefore, we introduced the weighting factors W_h and W_s. We always applied the values $W_h = 20$ and $W_s = 1$.

Now we are able to present the MIP:

$$\min W_h \sum_{e \in E} N_e y_e + W_s \left(\sum_{e \in E} N_e u_e + \sum_{s \in S} \sum_{t \in T'} v_{st} + \sum_{s \in S} \sum_{d} w_{sd} \right)$$

$$\sum_{e\in E_s|t\in T_e}\sum_{r\in R_e} z_{ert} \leq 1 \qquad \forall s\in S, \forall t\in T, \tag{13}$$

$$\sum_{e\in E|r\in R_e \& t\in T_e} z_{ert} \leq 1 \qquad \forall r\in R, \forall t\in T, \tag{14}$$

$$\sum_{r\in R_e}\sum_{t\in T_e} z_{ert} + y_e = 1 \qquad \forall e\in E, \tag{15}$$

$$\sum_{t\in T_e|t<t'}\sum_{r\in R_e} z_{ert} \geq \sum_{t\in T_f|t\leq t'}\sum_{r\in R_f} z_{frt} \qquad \forall e,f\in E|p_{ef}=1, \forall t'\in[1,44], \tag{16}$$

$$\sum_{t\in\{8,17,26,35,44\}}\sum_{r\in R_e} z_{ert} - u_e \leq 0 \qquad \forall e\in E, \tag{17}$$

$$\sum_{t=t'}^{t'+2}\sum_{e\in E_s|t\in T_e}\sum_{r\in R_e} z_{ert} - v_{st'} \leq 2 \qquad \forall s\in S, \forall t'\in T', \tag{18}$$

$$\sum_{t=9*d}^{9*d+8}\sum_{e\in E_s|t\in T_e}\sum_{r\in R_e} z_{ert} - M w'_{sd} \leq 0 \qquad \forall s\in S, \forall d\in D, \tag{19}$$

$$\sum_{t=9*d}^{9*d+8}\sum_{e\in E_s|t\in T_e}\sum_{r\in R_e} z_{ert} + w_{sd} \geq 2w'_{sd} \qquad \forall s\in S, \forall d\in D, \tag{20}$$

$$z_{ert} \in \{0,1\} \qquad \forall e\in E, \forall r\in R_e, \forall t\in T_e, \tag{21}$$

$$y'_e \geq 0 \qquad \forall e\in E, \tag{22}$$

$$u_e \geq 0 \qquad \forall e\in E, \tag{23}$$

$$v_{st} \geq 0 \qquad \forall s\in S, \forall t'\in T', \tag{24}$$

$$w_{sd} \geq 0 \qquad \forall s\in S, \forall d\in D, \tag{25}$$

$$w'_{sd} \in \{0,1\} \qquad \forall s\in S, \forall d\in D \tag{26}$$

Constraint (13) ensures that no student has to attend more than one event at a time. Only one event can be put into each room on any timeslot. This is enforced by constraints (14). Constraint (15) imposes that every event is assigned to at most one timeslot and one room. If there exists a precedence relation between two events, they should be scheduled in the correct order during the week. This is fulfilled by constraint (16). Constraint (17) accounts for the soft constraint that no student has to attend an event at the last time slot of a day. Students do not have to attend more than two events consecutively. This is taken into account by constraint (18). Constraints (19) and (20) together account for that a student is not assigned to one event on a certain day. Note that all decision variables except for the z-variables are elements of $\mathbb{R}^+$. They are automatically put on zero or one, because of the binary z-variables.

4.2 The improvement heuristic

The introduced MIP is too big to solve in a reasonable amount of time, but it can be used to fill up a partial solution. A partial solution is represented by a set E^p of events that have

been assigned to a timeslot. The timeslot to which an event $e \in E^p$ is assigned is denoted by τ_e. The events in E^p have not been assigned to a room, but it is known that a feasible room assignment exists for those events. Given a partial solution, the number of decision variables in MIP is reduced a lot by setting $\forall e \in E^p, \forall t \in T \setminus \tau_e : z_{ert} = 0$. If the set E^p contains enough events, then MIP is able to reassign each event $e \in E^p$ to a room and each events $f \in E \setminus E^p$ to a room and timeslot within a small computation time.

Given the solution constructed by the LP-based heuristic we try to find better solutions by solving MIP with input a partial solution derived from the known valid solution. We denote MIP with input a partial solution represented by E^p with $MIP(E^p)$. An advantage of using an integer program is that always a solution is found with quality as good as the currently best known solution.

We first describe what the number of soft constraint violations $NSCV(e)$ of an event e is. This number is determined for all events assigned to a timeslot and room in the current valid solution. It is counted as follows:

1. If τ_e is a last timeslot of the day, then $NSCV(e)$ is increased by N_e.
2. If e has a student for whom e is the only event on a day, then $NSCV(e)$ is increased by one.
3. If a student has three events in a row and one of the events is e, then $NSCV(e)$ is increased by one.

Given the current valid solution we have to remove events from the set E^p to construct a partial solution. It seems obvious to remove events which a high number of soft constraint violations. The problem then is that it is almost impossible to assign the removed events into the partial solution on different timeslots than in the old solution. Therefore the improvement heuristic chooses one event e^* from E to remove from E^p. The other events removed from E^p are events that increase the probability that a new solution is found with e^* assigned to a timeslot with lower soft constraint violations. These are for example events assigned to the same room or events colliding with e^* assigned to a timeslot $t \in T_{e^*}$.

The heuristic always starts by choosing an event e^* that is not assigned in the given valid solution. If the current solution is feasible, then event e with maximum $NSCV(e)$ is chosen as e^*. To prohibit that always the same event e^* is chosen, a list L of length 15 is used that contains the events that were most recently used as event e^*.

The parameter $c(t, e^*)$ is the number of events $e \in E^p$ with $\tau_e = t$ that collide with e^*. With T^c we denote the set of timeslots for which all assigned events are removed from E^p. The timeslots in T^c are feasible timeslots for e^* with the largest probability that e^* can be assigned to one of those timeslot in a better solution. These are the timeslots with the minimum number of colliding events with e^*. If this is equal the timeslot with the maximum number of soft constraint violations is chosen.

The improvement heuristic goes as follows:

1. Initialize $L = \emptyset$ and $NOT = 5$.
2. $E^p = E \setminus L$.
3. Select $e^* \in E^p : e^*$ is not assigned or $e^* = \max_{f \in E \setminus L} NSCV(f)$.
4. $T^c = T_{e^*}$.
5. Delete timeslots from T^c for which event e^* is not feasible, because of precedence constraints.
6. Reduce T^c to NOT timeslots with $\min_{t \in T^c} c(t, e^*)$ or $\max_{t \in T^c} \sum_{f \in E | t = \tau_f} NSCV(f)$.
7. $\forall f \in E^p$: If $\tau_f \in T^c$, then remove f from E^p.
8. If $|R_{e^*}| \leq 1$, then $\forall f \in E^p$ with $R_{e^*} = R_f$: remove f from E^p.

9. $\forall f \in E^p$: If $c(\tau_f, e^*) <= 2$ and f collides with e^*, then remove f from E^p.
10. Solve $MIP(E^p)$.
11. Add e^* to L and delete the last element of L.
12. If no improvement found for 15 iterations, then $NOT = NOT + 1$.
13. If optimal solution not found and there is computation time left, then go to step 2.

If e^* has only one possible room, then step 8 removes the events from E^p that also have only one possible room which is the same as the room of e^*. To enlarge the probability that an event e^* is assigned to a different timeslot, the events colliding with e^* in timeslots feasible for e^* and that do not have more than two colliding events with e^* are removed from E^p in step 9. The solution found in step 10 is always as good as the old one, therefore the new solution is always accepted.

5 Computational results

We start with presenting the results of the LP-based heuristic described in Sect. 3. To solve RMP, CPLEX 10.0 is used. The results of the heuristic are given in Table 1 for all 24 instances. The column with head *c.t.*(*s*) gives the computation time of the heuristic. The column with *dtf* presents the distance to feasibility. The next column represents the number of

Table 1 Computational results of the LP-based heuristic

I	$\|E\|$	$\|R\|$	$\|S\|$	*c.t.*(s)	*dtf*	3 cons	1 a day	*eod*	s.c.
1	400	10	500	50	0	1391	20	640	2051
2	400	10	500	55	0	1429	19	677	2125
3	200	20	1000	5	0	339	595	0	934
4	200	20	1000	5	0	417	581	81	1079
5	400	20	300	34	0	823	14	186	1023
6	400	20	300	38	0	1002	24	239	1265
7	200	20	500	6	0	291	287	73	651
8	200	20	500	4	0	244	259	110	613
9	400	10	500	58	0	1594	24	802	2420
10	400	10	500	73	0	1356	17	692	2065
11	200	10	1000	5	0	343	566	167	1076
12	200	10	1000	5	0	462	573	0	1035
13	400	20	300	33	0	789	8	262	1059
14	400	20	300	44	0	898	30	274	1202
15	200	10	500	5	0	316	325	74	715
16	200	10	500	4	0	252	285	74	611
17	100	10	500	7	0	295	6	0	301
18	200	10	500	5	0	860	36	233	1129
19	300	10	1000	24	0	489	525	485	1499
20	400	10	1000	56	0	734	605	297	1636
21	500	20	300	75	0	783	21	215	1019
22	600	20	500	331	18	1414	14	756	2184
23	400	20	1000	48	0	2791	28	654	3473
24	400	20	1000	35	0	711	666	233	1610

Table 2 Computational results after the improvement heuristic

I	$\|E\|$	$\|R\|$	$\|S\|$	*dtf*	3 cons	1 a day	*eod*	s.c.
1	400	10	500	0	1033	18	585	1636
2	400	10	500	0	903	14	717	1634
3	200	20	1000	0	102	253	0	355
4	200	20	1000	0	237	407	0	644
5	400	20	300	0	342	2	181	525
6	400	20	300	0	469	7	164	640
7	200	20	500	0	0	0	0	0
8	200	20	500	0	113	128	0	241
9	400	10	500	0	1095	12	782	1889
10	400	10	500	0	1080	19	578	1677
11	200	10	1000	0	261	354	0	615
12	200	10	1000	0	236	292	0	528
13	400	20	300	0	309	2	174	485
14	400	20	300	0	459	12	268	739
15	200	10	500	0	136	194	0	330
16	200	10	500	0	86	159	15	260
17	100	10	500	0	33	2	0	35
18	200	10	500	0	443	8	52	503
19	300	10	1000	0	211	309	443	963
20	400	10	1000	0	379	495	355	1229
21	500	20	300	0	481	6	183	670
22	600	20	500	0	1172	14	770	1956
23	400	20	1000	0	1899	16	453	2368
24	400	20	1000	0	389	435	121	945

times a student is assigned to three consecutive events. The number of times a student has only one event on a day is given in the column '1 a day' and the number of students having an event on the last timeslot of the day is given in the column *eod*. The right most column gives the total number of soft constraint violations.

The LP-based heuristic is able to assign all events to a timeslot and a room for 23 out of 24 instances. For instance 22 one event with 18 students could not be assigned. The number of soft constraint violations is quite large, but we took them into account only in a minor way. So we can conclude that our LP-based heuristic succeeds in fulfilling our main goal, which was providing a solution with distance to feasibility as small as possible.

For a fair comparison of computational results the organizers provided a benchmark program to determine the maximum allowed computation time on a machine. This allowed computation time on the machine we used was 429 seconds. Looking at Table 1 it can be seen that for most instances a lot of computation time is left to improve the solution found. The improvement heuristic is run until this allowed computation time is fully used or until an optimal solution has been found. Table 2 presents the computational results if the improvement heuristic is used with the solution found by the LP-based heuristic as input.

After the improvement heuristic also instance 22 has a distance to feasibility of zero. One of the first $MIP(E^p)$ that was solved found a solution with distance to feasibility zero. The

Table 3 Comparison with solutions of the five finalists

I	*dtf*1	*dtf*2	*dtf*3	*dtf*4	*dtf*5	*dtf*TUe	sc1	sc2	sc3	sc4	sc5	scTUe
1	8.3	0.0	0.0	445.5	22.2	0	647.6	883.4	1730.5	1071.2	1927.0	1636
2	30.4	*0.0	0.0	335.8	171.9	0	884.6	*1252.7	1913.6	677.9	2201.8	1634
3	0.0	0.0	0.0	0.0	0.0	0	529.9	237.3	389.7	732.6	333.9	355
4	0.0	0.0	0.0	0.0	0.0	0	683.2	370.0	480.2	727.5	559.7	644
5	0.0	0.0	0.0	0.0	0.0	0	21.0	6.8	679.9	128.3	20.9	525
6	0.0	0.0	0.0	3.9	0.0	0	64.5	4.2	977.4	391.9	266.6	640
7	0.0	0.0	0.0	0.0	0.0	0	97.7	7.5	354.1	3.8	183.6	0
8	0.0	0.0	0.0	0.0	0.0	0	24.1	0.0	1.3	80.6	24.5	241
9	79.9	0.0	0.0	684.3	346.5	0	832.9	1868.6	2100.4	1080.5	2407.7	1889
10	8.1	+0.0	37.1	0.0	577.5	0	231.7	+555.0	2272.3	0.1	2319.0	1677
11	0.0	0.0	0.0	0.0	5.4	0	716.0	288.3	352.6	898.0	742.9	615
12	0.0	0.0	0.0	112.8	14.1	0	1046.8	352.7	616.4	1275.3	1293.1	528
13	0.0	0.0	0.0	6.5	0.0	0	102.6	128.3	911.1	478.6	475.6	485
14	0.0	0.0	0.0	0.0	0.0	0	0.4	4.1	983.5	97.0	407.9	739
15	0.0	0.0	0.0	0.0	0.0	0	460.6	93.1	310.6	142.7	268.2	330
16	0.0	0.0	0.0	0.0	0.0	0	251.8	17.1	5.8	131.2	178.4	260
17	0.0	0.0	0.0	0.0	0.0	0	19	4.9	9.8	116.4	106.2	35
18	0.6	0.0	0.0	0.0	0.0	0	39.4	14.1	339.9	264.8	314.3	503
19	348.7	−0.0	0.0	89.7	755.1	0	1838	−405.4	2081	233.1	2314	963
20	16.4	0.0	0.0	771.2	0.0	0	1288	505	640.5	2383	919.3	1229
21	0.0	0.0	0.0	14.3	0.0	0	3.6	27.1	876.3	326.6	336.8	670
22	0.0	=0.0	0.0	0.0	0.0	0	0.0	=550.8	1839	82.7	1593.7	1956
23	3.4	0.0	0.0	239.8	0.0	0	573.9	330.5	1043	1274	701.3	2368
24	13.1	0.0	0.0	0.0	0.0	0	911.1	124.2	963.4	129.2	518	945

*8 out of the 10 runs gave a valid solution. The average score of these 8 solutions is given

+3 out of the 10 runs gave a valid solution. The average score of these 3 solutions is given

−2 out of the 10 runs gave a valid solution. The average score of these 2 solutions is given

=9 out of the 10 runs gave a valid solution. The average score of these 9 solutions is given

number of soft constraint violations decreased a lot. Especially shortly after the start of the improvement heuristic a lot of large improvements are found.

To get an impression of how good the total algorithm performs, we compare our results with the results of the five finalists of the competition. For these five finalists the organizers did 10 runs with different random seed for every instance. This makes it very difficult to compare the results, because our heuristic is deterministic. Table 3 shows the average of the 10 runs for each of the five finalists (1–5) and the results of our heuristic (TUe).

Our heuristic provides a solution with distance to feasibility of zero for all 24 instances. In comparison with the average distance to feasibility of the finalists our algorithm performs well. Especially if one takes into account that for most instances the distance to feasibility is found in a small computation time. Looking at the number of soft constraint violations we are competitive with the finalists. It clearly depends on the instance how well we perform. Because it is not known which soft constraints are violated by the finalists, we can also not say much about which algorithm can deal the best with a certain type of constraint.

6 Conclusions

We presented a heuristic that constructs a feasible solution for the post-enrolment course timetabling problem of the second international timetabling competition. The heuristic fixes a set of events for a timeslot given a feasible solution of an LP-relaxation which we find by using column generation. Our construction heuristic is able to assign all the events to a timeslot and a room without violating any of the hard constraints in 23 out of 24 instances of the competition. In comparison with the five finalists of the competition our heuristic performs very well in generating a solution with distance to feasibility of zero. Most finalists have one or more instances where their algorithm does not find a feasible solution

A second step to decrease the number of soft constraint violations of the solution is presented. Every iteration a set of event is released and assigned again by solving an integer program to optimality. Looking at the number of soft constraint violations our total heuristic is competitive with the finalists.

References

Atsuta, M., Nonobe, K., & Ibaraki, T. (2008). *ITC-2007 Track2: an approach using general CSP solver*. Submission to ITC2007 track 2.

Borndörfer, R. M., Grötschel, & Löbel, A. (2003). Duty scheduling in public transit. In W. Jäger & H.-J. Krebs (Eds.), *Mathematics-key technology for the future* (pp. 653–674). Berlin: Springer.

Cambazard, H., Hebrard, E., O'Sullivan, B., & Papadopoulos, A. (2008). Local search and constraint programming for the post-enrolment course timetabling problem. In *Proceedings of the conference on the practice and theory of automated timetabling (PATAT 2008)*, Montreal, Canada, 2008.

Carter, M. W., & Laporte, G. (1998). Recent developments in practical course timetabling. In E. K. Burke & M. W. Carter (Eds.), *Lecture notes in computer science: Vol. 1408. Practice and theory of automated timetabling II*. Berlin: Springer.

Chiarandini, M., Fawcett, C., & Hoos, H. H. (2008). A modular multiphase heuristic solver for post enrolment course timetabling. In *Proceedings of the conference on the practice and theory of automated timetabling (PATAT 2008)*. Montreal, Canada, 2008.

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (2001). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In C. C. Ribiero & P. Hansen (Eds.), *Essays and surveys in metaheuristics* (pp. 309–324). Boston: Kluwer.

Desrosiers, J., & Lübbecke, M. E. (2005). A primer in column generation. In Desaulniers, G., J. Desrosiers, & M. M. Solomon (Eds.), *Column generation*, Chap. 1 (pp. 1–32). New York: Springer.

Hutter, F., Babic, D., Hoos, H. H., & Hu (2007). Boosting verification by automatic tuning of decision procedures. In *Formal methods in computer aided design* (FMCAD).

International Timetabling Competition (ITC) (2007). http://www.cs.qub.ac.uk/itc2007.

Kostuch, P. (2004). The university course timetabling problem with a three-phase approach. In E. K. Burke & M. Trick (Eds.), *Lecture notes in computer science: Vol. 3616. Practice and theory of automated timetabling V* (pp. 109–125). Berlin: Springer.

Lewis, R., Paechter, B., & McCollum, B. (2007). *Post enrolment based course timetabling: a description of the problem model used for track two of the second international timetabling competition* (Technical Report). Cardiff University.

Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, *53*(6), 1007–1023.

Mayer, A., Nothegger, C., Chwatal, A., & Raidl, G. R. (2008). Solving the post enrolment course timetabling problem by ant colony optimization. In *Proceedings of the conference on the practice and theory of automated timetabling (PATAT 2008)*, Montreal, Canada, 2008.

Müller, T. (2008). ITC2007 solver description: a hybrid approach. In *Proceedings of the conference on the practice and theory of automated timetabling (PATAT 2008)*, Montreal, Canada, 2008.

Rossi-Doria, O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L. M., Knowles, J. D., Manfrin, M., Mastrolilli, M., Paechter, B., Paquete, L., & Stützle, T. (2002). A comparison of the performance of different metaheuristics on the timetabling problem. In E. K. Burke & P. De Causmaecker (Eds.), *Lecture notes in computer science: Vol. 2740. Practice and theory of automated timetabling IV* (pp. 329–351). Berlin: Springer.

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