Our solution approach is a heuristic that assigns events to timeslots and rooms given a feasible, not necessarily optimal, solution of a linear program. We use column generation to find a feasible solution of the LP. First we define the necessary parameters, then we give the description of the column generation procedure and in the end we describe the heuristic.

Parameters

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 of the set R of rooms. We also have a set S of students, and every student s has a set E_s of events that he is 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. Every room $r \in R$ has a set of features F_r and a specific seating capacity C_r. 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 larger or equal than the number of students participating ($N_e \leq C_r$). We define the set R_e as the set of rooms to which event e can be assigned. Precedences between events are modeled with the parameter p_{ef}. It has value one if e is the predecessor of f.

An important parameter is c_e, this is the number of events colliding with event e. Two events are *colliding* if they can not be scheduled at the same timeslot. This arises if they have at least one 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 K as the set of slot-schedules. A *slot-schedule* $k \in K$ is characterized by a timeslot t_k and a set of events assigned to this timeslot. For all slot-schedules $k \in K$ and all events $e \in E$ we introduce the parameter a_{ke}, which is one if event e is in slot-schedule k and zero otherwise. A slot-schedule k is feasible if:
- The events in the slot-schedule are not colliding.
- $t_k \in T_e$ for all events e in the slot-schedule.
- If an event e is assigned to a room $r \in R_e$.
- Only one event is assigned to each room.

The master problem (MP)

MP has three types of decision variables:
- $\forall k \in K : x_k$ is the number of times slot-schedule k is assigned.
- $\forall e \in E : y_e = 0$ if event e is assigned, stricty positive otherwise.
- $\forall e, f \in E :$ If $p_{ef} = 1$, then $z_{ef} \in \mathbb{Z}^+$ is a variable that indicates whether the precedence constraint between e and f is fullfilled. A feasible solution has z_{ef} is zero.

MP is the following linear programming model:

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

$$y_e + \sum_{k \in K} a_{ke} x_k \geq 1 \qquad \forall e \in E \tag{1}$$

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

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

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

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

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

Constraint (1) ensures that all events are assigned. For every timeslot at most one slot-schedule can be selected, this is enforced by constraint (2). A precedence constraint between two events is fulfilled by constraint (3).

In the initial set of columns we generate for all n events a column with only the corresponding y_e variable on one. We do the same for all the precedence constraints by setting only the corresponding z_{ef} variable to one. For all timeslots in T we choose an empty slot-schedule k for which we take a column with only a one for the coefficient of x_k. To solve MP we start with $t = 0$ and solve pricing problems in order of increasing value of t to extend the set K. If for all timeslots in T no new slot-schedules are found, then the column generation procedure is stopped, independent whether an optimal solution is found or not. The RMP is solved with CPLEX 10.0.

The pricing problem

The pricing problem determines feasible slot-schedules that can be added to the restricted master problem (RMP). It has as input a timeslot t, a set E^p of events that can be scheduled and the shadowprices of RMP. These are given by α_e, β_t and γ_{ef}. The decision variable y'_e is one if event e is assigned and zero otherwise. Variable y_{er} is one if event e is assigned to room r and zero otherwise. The pricing problem is formulated as follows:

$$\max \sum_{e \in E^p} \alpha_e y'_e + \beta_t + \sum_{e,f \in E^p | p_{ef} = 1} \gamma_{ef} t (y'_f - y'_e)$$

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

$$\sum_{e \in E^p \cap E^s} y'_e \leq 1 \qquad \forall s \in S \tag{8}$$

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

$$y_e + y_f \leq 1 \qquad \forall e, f \in E^p | p_{ef} = 1 \tag{10}$$

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

$$y_e \in \{0, 1\} \qquad \forall e \in E^p \tag{12}$$

Solving the pricing problem with an IP solver takes too much computation time. Therefore, we apply a greedy heuristic that generates a set K^p of 30 feasible slot-schedules. We define obj_e as the value that event e adds to the above given objective function. The greedy heuristic goes as follows:

1. Select event e_f with maximum value of obj_e. If $obj_e = obj_f$ for events e and f, then take the event with minimum c_e.
2. Select the room $r \in R_{e_f}$ with the most events $e \in E^p$ with $R_e = \{r\}$. If this is equal for two rooms, then take the room that is in the minimum number of sets R_e with $e \in E^p$.
3. As long as there are rooms and events left, go back to 1.

To generate 30 different slot-schedules we always delete the first selected event e_f from E^p. Note that we only add a slot-schedule $k \in K^p$ as a column to RMP if the reduced costs of the slot-schedule are larger than the average reduced costs over the last 50 added slot-schedules.

Heuristic Based on LP solution

We define E_c as the set of events that still have to be scheduled and T_c as the set of timeslots to which a slot-schedule still can be assigned. Our heuristic is the following algorithm:

1. Initialize $T_c = T \backslash \{8, 17, 26, 35, 44\}$ and $E_c = E$.
2. Solve column generation procedure $\rightarrow K$.
3. For all generated slot-schedules $k \in K$, determine $obj_k = \sum_{e \in E} a_{ke} w_e$.
4. Assign the events in the slot-schedule k with maximum obj_k to t_k and delete them from E_c.
5. $T_c = T_c \backslash t_k$.
6. If $|T_c| > 0$ and $|E_c| > 0$, then go to step 2.
7. If $|E_c| > 0$, then solve an integer program to optimality (with CPLEX 10.0) to assign as many as possible of the events in E_c to the timeslots in $\{8, 17, 26, 35, 44\}$.

The value of the weighting factor w_e for an event e depends on the values of c_e, on the timeslot t_k in combination with the precedence constraints and on the number of rooms to which an event can be assigned.

The integer programming model that we solve in step 7, has decision variable z_{ert} which is one if event e is assigned to room r on timeslot t and zero otherwise. The variable z_{ert} is only defined for the triples (e, r, t) if $t \in T_e$ and $r \in R_e$. Our objective is to maximize the number of students assigned. The IP is:

$$\max \sum_{e \in E} (N_e (\sum_{r \in R_e} \sum_{t \in T_e} z_{ert}))$$

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

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

$$\sum_{r \in R_e} \sum_{t \in T_e} z_{ert} \leq 1 \qquad \forall e \in E \tag{15}$$

$$\sum_{t \in T_j | t < t'} \sum_{r \in R_j} z_{jrt} \geq \sum_{t \in T_k | t \leq t'} \sum_{r \in R_k} z_{krt} \qquad \forall j, k \in E | p_{jk} = 1, \forall t' \in [1, 44] \tag{16}$$

$$z_{ert} \in \{0, 1\} \qquad \forall e \in E, \forall r \in R_e, \forall t \in T_e \tag{17}$$

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 of the timeslots and one of the rooms. 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).