An LP-based Heuristic for the post Enrolment Course Timetabling problem of the ITC

John van den Broek and Cor Hurkens

Department of Mathematics and Computer Science,
Eindhoven University of Technology,
Den Dolech 2, 5600 MB Eindhoven, The Netherlands
j.j.j.v.d.broek@tue.nl, wscor@win.tue.nl

Abstract. We present a deterministic heuristic for the post enrolment course timetabling problem of the ITC. The heuristic is based on an LP-solution constructed with column generation. We get an integer solution by fixing a column one at a time. Our results are compared with the results of the five finalists.

1 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. There is 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 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.

The constraints in the timetabling problem can be divided into soft and hard constraints. 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 timeslot.
4. An event e can only be assigned to a timeslot $t \in T_e$.
5. Events have to be scheduled in the prescribed order during the week.

The soft constraints that have to be satisfied are:

1. Events should not be assigned to the last timeslots of a day.
2. Students should not have to attend three or more events in successive timeslots on the same day.

3. Students should not be required to attend only one event a day.

A *valid timetable* is one in which there are no hard constraint violations. 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. The most important goal is finding a valid solution with minimum *distance to feasibility (dtf)*, which is the number of students that require an unplaced event. If two valid solutions have the same distance to feasibility, then the solution with the minimum number of violated soft constraints is preferred.

2 Heuristic Based on Column Generation

The heuristic 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.

We define c_e as the number of events colliding with event e. Two events are *colliding* if they can not be scheduled on 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 E_k of events assigned to this timeslot. A slot-schedule k is feasible if:
- all $e \in E_k$ are not colliding.
- $t_k \in T_e, \forall e \in E_k$.
- Event e is assigned to a room $r \in R_e, \forall e \in E_k$.
- At most one event is assigned to each room.

We may consider the complete set of possible slot-schedules and model the problem as an integer linear program with binary decision variable x_k which indicates whether slot-schedule k is used or not. We cannot handle such a big ILP, therefore we solve the LP-relaxation (not necessarily to optimality) using column generation. For all events and precedence constraints we generate a slack variable. The initial set of columns contains a column with only a 1 for the coefficient of these slack variables. For all timeslots in T we generate a slot-schedule k with $E_k = \emptyset$, for which we take a column with only a 1 for the coefficient of x_k.

The pricing problem determines feasible slot-schedules that can be added to the set K. The pricing problem p has as input a timeslot t, a set E^p of events that can be scheduled in p and the shadowprices after solving the restricted master problem (RMP). We apply a greedy heuristic that generates a set K^p of 30 feasible slot-schedules. A slot-schedule $k \in K^p$ is added as a column to RMP if the reduced costs of the slot-schedule are larger than zero and larger than the average reduced costs over the 50 most recently added slot-schedules.

The column generation procedure starts with $t = 0$ and solves pricing problems in order of increasing value of t to extend the set K. The column generation

procedure stops when for all timeslots no slot-schedules are found that are added to RMP. This is independent of whether an optimal solution is found or not.

To get an integer feasible solution of the master problem we apply a heuristic that determines the best column of all generated columns in RMP. The best column is mainly determined by the sum of all c_e of the events in the corresponding slot-schedule. The events of this best column are assigned to the corresponding timeslot, by fixing its corresponding x-variable to 1. Given this assignment of events the LP-relaxation is solved again. We repeat this until for every timeslot a slot-schedule is determined or until all events have been scheduled.

Note that timeslots at the end of a day are never chosen. After all timeslots, except for those, have a slot-schedule assigned, the resulting events are assigned to the last timeslots of the day by solving another integer programming model to optimality.

3 Comparison of our results with the finalists

For the five finalists of the competition the organizers did 10 runs for every instance. Table 1 shows the average of the results for each of the five finalists $(1-5)$ and the results of our heuristic (TUe). The columns with *sc* contain the average number of soft constraint violations. We have a deterministic algorithm, therefore it is difficult to compare the results.

Our heuristic gives a solution with a distance to feasibility of zero for all 16 instances. In comparison with the average distance to feasibility of the finalists our algorithm performs well.

We did not focus on minimizing soft constraint violations at all, so it is not a surprise that our heuristic performs worse with respect to the number of soft constraint violations. Future research is focused on reducing the number of soft constraint violations by tailoring the column generation procedures.

References

1. http://www.cs.qub.ac.uk/itc2007

Table 1. Comparison with solutions of the five finalists

I	dtf1	dtf2	dtf3	dtf4	dtf5	dtfTUe	sc1	sc2	sc3	sc4	sc5	scTUe
1	8.3	0.0	0.0	445.5	22.2	0.0	647.6	883.4	1730.5	1071.2	1927.0	2424
2	30.4	*0.0	0.0	335.8	171.9	0.0	884.6	1252.7	1913.6	677.9	2201.8	2322
3	0.0	0.0	0.0	0.0	0.0	0.0	529.9	237.3	389.7	732.6	333.9	1626
4	0.0	0.0	0.0	0.0	0.0	0.0	683.2	370.0	480.2	727.5	559.7	1584
5	0.0	0.0	0.0	0.0	0.0	0.0	21.0	6.8	679.9	128.3	20.9	1263
6	0.0	0.0	0.0	3.9	0.0	0.0	64.5	4.2	977.4	391.9	266.6	1369
7	0.0	0.0	0.0	0.0	0.0	0.0	97.7	7.5	354.1	3.8	183.6	708
8	0.0	0.0	0.0	0.0	0.0	0.0	24.1	0.0	1.3	80.6	24.5	758
9	79.9	0.0	0.0	684.3	346.5	0.0	832.9	1868.6	2100.4	1080.5	2407.7	2696
10	8.1	+0.0	37.1	0.0	577.5	0.0	231.7	555.0	2272.3	0.1	2319.0	2389
11	0.0	0.0	0.0	0.0	5.4	0.0	716.0	288.3	352.6	898.0	742.9	1652
12	0.0	0.0	0.0	112.8	14.1	0.0	1046.8	352.7	616.4	1275.3	1293.1	1818
13	0.0	0.0	0.0	6.5	0.0	0.0	102.6	128.3	911.1	478.6	475.6	1436
14	0.0	0.0	0.0	0.0	0.0	0.0	0.4	4.1	983.5	97.0	407.9	1275
15	0.0	0.0	0.0	0.0	0.0	0.0	460.6	93.1	310.6	142.7	268.2	943
16	0.0	0.0	0.0	0.0	0.0	0.0	251.8	17.1	5.8	131.2	178.4	893

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

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