DM87 - Scheduling, Timetabling and Routing

Course Timetabling

May 30, 2008

Niels H. Kjelsen & Jacob Aae Mikkelsen

Contents

1 Introduction and Problem Description

This report documents part of the exam in the course DMP87 - Scheduling, Timetabling and Routing, on IMADA, SDU, spring 2008. The topic is course timetabling, in general and for specific IMADA instances. The original assignment indicated that exact methods could solve some of the instances, but not all, so heuristic methods would be needed. This turned out not being necessary for the instances posed, so an extra task was added. By then, we already had simulated annealing implemented, so this is also described in the report.

The extra task chosen consist of multiobjective analysis, characterizing the Pareto frontier of solutions, considering each soft constraint as an objective. Furthermore, a study of how the solution would behave if an extra room was added or removed, and a study of how the solution would become, if all mandatory courses could be rescheduled. Finally, a study of how many courses a student in the IMADA instances could enroll before a valid schedule no longer exists.

In the course timetabling problem considered, the following two kinds of constraints apply:

Hard constraints

1. Only one lecture is assigned in a room in a timeslot.
2. No assignment to a room in a timeslot if it is unavailable.
3. No teacher is teaching two courses in the same timeslot, or in a timeslot the teacher is unavailable.
4. No student has to attend more than one lecture in a timeslot.

Soft constraints

1. Lectures should not be scheduled in the first or last timeslot in a day.
2. Two lectures in the same course should not be placed on the same day or two consecutive days.
3. Each student should have at most 3 lectures in a day.

In the instances, a set of rooms are included, which are free to use. It is possible to schedule a lecture in a rooms not in this set, using a dummy room.

We have two definitions of a timetable:

Feasible All lectures are scheduled in the rooms available (no dummy room used) and no hard constraints are violated.

Valid All lectures are scheduled, no hard constraint are violated, but a dummy room is used.

The exact evaluation of a schedule is described in the project description, but in summary it is a tuple (DISTANCE TO FEASIBILITY , SOFT COST), where DISTANCE TO FEASIBILITY is the number of students attending the lectures scheduled in dummy rooms, and SOFT COST is violations of the soft constraints.

2 Solution Design

2.1 Integer Programming Formulation

Three versions of the integer programming formulation has been made. The first guarantees an optimal schedule, if a feasible one exist. Room assignments must be handled afterwards. The next, which also includes the room assignment (and the extra constraint with courses only suited for some of the rooms) Has a much larger decision variable, but will find the optimal schedule, even if the use of dummy rooms is necessary. The third is like the first, modified slightly for the multicriteria task. All versions use a JAVA program to do pre- and post-processing, Zimpl as modelling language and Scip as solver, alll connected in a Bash script for easy execution.

Definitions of sets and vectors

We define the following sets and vectors, common for the IP formulations: E is the set of elective courses,M the set of mandatory courses, T is the set of timeslots, R is the set of rooms and S is the set of students. BT represents the set of timeslots located in the first or last period of a day day. $L[c]$ is a vector defining the number of lectures in the course c, $En[c]$ defines the number of students enrolled in the course c, and $Takes$ is a set of pairs (s, c), for which student s takes course c and TE is a set of pairs (s, c), for which student s takes elective course c. $Teaches$ is a set of pairs (c, t), for which teacher s teaches course c, and TU is a set of pairs (t, i), for which teacher t is unavailable in timeslot i. $m[s]$,$t[s]$,$w[s]$,$h[s]$ and $f[s]$ are decision vectors telling if student s has more than 3 lectures on any of the days Monday to Friday. $r[t]$ is a decision vector telling if a dummy room is used in timeslot t, and $d[c]$ is a decision vector telling if the minimum distance of course c is violated.

Version 1: Optimal if a feasible schedule exist

In this version, the main decision variable $x[c, t]$ only decides if course c has a lecture in timeslot t. $Ra[t]$ is a vector defining the number of rooms available in timeslot t.

The objective is to minimize:

$$\sum_{t\in BT}\sum_{c\in E} x[c,t]\cdot En[c] + \sum_{s\in S} m[s]+t[s]+w[s]+t[s]+f[s] + \sum_{c\in E} d[c]\cdot En[c] + \sum_{t\in T} 1000\cdot r[t] \quad (2.1)$$

Subject to the following constraints:

$$\sum_{c,t \in E \times T} x[c,t] = L[c] \qquad \forall c \in E \tag{2.2}$$

Which ensures the number of lectures in each course is correct.

$$\Big(\sum_{c,t \in E \times T} x[c,t]\Big) - Ra[t] <= r[t] \qquad \forall t \in T \tag{2.3}$$

Which ensures that if we use more rooms than we have available in a timeslot, then $r[t]$ is set to one.

$$x[c1,t] + x[c2,t] \leq 1 \qquad \forall t \in T, \forall (teacher, c1), (teacher, c2) \in Teaches, c1 \neq c2 \tag{2.4}$$

Which ensures that no teacher must teach two lectures in the same timeslot.

$$x[c,t] = 0 \qquad \forall (teacher, t) \in TU, \forall (teacher, c) \in Teaches \tag{2.5}$$

Which ensures that no teacher teaches in a timeslot, where he's unavailable.

$$x[c1,t] + x[c2,t] \leq 1 \qquad \forall t \in T, \forall (s, c1), (s, c2) \in TE, c1 \neq c2 \tag{2.6}$$

Which ensures that no student is scheduled to attend two elective courses at the same time.

$$x[c1,t] = 0 \qquad \forall (c2,t) \in M, \forall c1 \in E, \forall (s1, c1) \in TE, \forall (s2, c2) \in Takes, s1 = s2 \tag{2.7}$$

Which ensures that no student is scheduled to attend a mandatory and an elective course in the same timeslot.

$$(500 + \sum_{(s,c) \in TE} \sum_{t \in Mon} x[c,t])/(503 - manMonday[s]) \leq 1 + m[s] \qquad \forall s \in Students \tag{2.8}$$

Which lets $m[s]$ be 0 if a student has at most 3 lectures on Monday including mandatory lectures, and 1 otherwise. Such a constraint also exist for the other four days.

$$\sum_{t \in MonTue} x[c,t] <= 1 + d[c] \qquad \forall c \in E \tag{2.9}$$

Which ensures $d[c]$ is set to 1 if there is scheduled two lectures in the same course on Monday and Tuesdays. Otherwise $d[c]$ is set to 0. Likewise, there are constraints for the pair of days: Tuesday and Wednesday, Wednesday and Thursday, and Thursday and Friday.

Since the maximum day distance is 4, this cannot be violated.

Version 2: Optimal, even if a feasible schedule doesn't exist

In this version, the main decision variable x now has a larger dimension, so $x[c,t,r]$ is 1 if course c has a lecture scheduled in timeslot t in room r, and 0 otherwise. This version also handles the extra constraint, that only a subset of the rooms are valid for each course. Most of the definitions from the former version also aplies here, further R represent the set of rooms, including a dummy room, named *dummy*. w is a large penalty for using a dummy room. RU is a set of tuples (r,t), for which room r is unavailable at timeslot t. CR is a list of tuples (c,r) for which course c is not allowed to be scheduled in room r.

The objective function to minimize is now:

$$\sum_{t\in T}\sum_{c\in E} x[c,t,dummy]\cdot En[c]\cdot w + \sum_{t\in BT}\sum_{c\in E}\sum_{r\in R} x[c,t,r]\cdot En[c] + \sum_{s\in S} m[s]+t[s]+w[s]+t[s]+f[s]+\sum_{c\in E} d[c]\cdot En[c] \tag{2.10}$$

Most of the constraints from the previous formulation are just extended with a sum of the rooms, except (2.3), which is replaced by the following two constraints:

$$\sum_{c\in E} x[c,t,r] \le 1 \qquad \forall t\in T, \forall r\in R \tag{2.11}$$

Which allows at most one lecture to be scheduled in a room in a timeslot.

$$x[c,t,r] = 0 \qquad \forall (r,t)\in RU, \forall c\in E \tag{2.12}$$

Which makes sure no lecture is scheduled in a room that is unavailable.

The final constraint in this version handles the extra task:

$$x[c,t,r] = 0 \qquad \forall t\in T, \forall (c,r)\in CR. \tag{2.13}$$

Note that this list is the inverted of the one given in the assignment.

Version 3: Used for multicriteria analysis

This version is very similar to version 1, except weight factors have come into place on the objective function, which now looks like:

$$a_1\cdot\sum_{t\in BT}\sum_{c\in E} x[c,t]\cdot En[c]+a_2\cdot\sum_{s\in S} m[s]+t[s]+w[s]+t[s]+f[s]+a_3\cdot\sum_{c\in E} d[c]\cdot En[c]+\sum_{t\in T} 1000\cdot r[t] \tag{2.14}$$

Solutions

The optimal solutions found using the integer programming approach is displayed in table 2.1

Trade-off between the objectives

If the tree soft constraint are considered as single objectives, they can be assigned a weight, and a possible trade-off between them can be examined.

	Version 1		Version 2	
Instance	Cost	Time (Min)	Cost	Time (Min)
E04	(0,84)	0.02	(0,84)	0.02
E05	(0, 83)	0.02	(0, 83)	0.02
E06	(0, 11)	0.02	(0, 11)	0.02
F05	(0, 44)	0.02	(0, 44)	0.02
F06	(0,79)	0.02	(0,79)	0.02
comp-2007-2-15	(0,156)	0.39	(0,156)	3.33
comp-2007-2-16	(0,197)	3.06	(0,197)	10.08
comp-2007-2-7	(0,372)	31.19	(0,372)	N/A.
comp-2007-2-8	(0,280)	3.00	(0,280)	15.37

Table 1: Results using integer programming with Zimpl and Scip

Let A represent the objective which corresponds to soft constraint 1, B soft constraint 2 and C represent 3. The objective can then be written as:

$$F = a_1 \cdot A + a_2 \cdot B + a_3 \cdot C \tag{2.15}$$

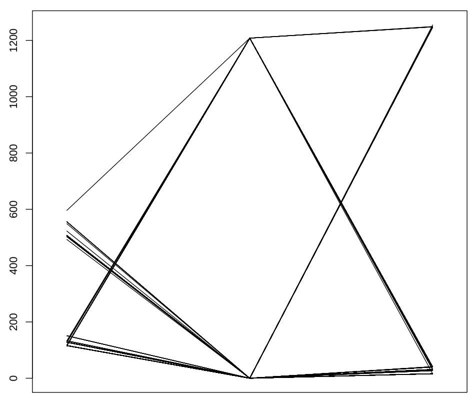
Where the a_i's are weights on each of the three constraints. If we let $l, m, n \in 0, .., 7$ then we can calculate the normalized a_i's as $\frac{l}{(l+n+m)}, \frac{m}{(l+n+m)}$ and $\frac{n}{(l+n+m)}$ (excluding $l = n = m = 0$). This Produces a number of different weights, and graphing them all (for the instance large-comp-2007-15), the result can be found in figure 1(a). This graph is difficult to interpret, so instead three experiments are performed: One of the soft constraints is fixed at weight 10, and the other two must share 10. This is done for each of the soft constraints, and the results can be found in figure 1(b), 1(c) and 1(d).

When we consider figure 1(b), half of the weight is used on objective A, and the red and grey line represent the two problems, where each of the other two objectives have weight zero. From this, there does not seem to be a trade off between the second and third soft constraint. When haft the weight is used on objective B, figure 1(c), there seems to be a trade-off between objective A and C. In figure 1(d) half the weight goes to objective C, and this leads to the same conclusion, there is a trade-off between A and C. In summary, the minimal day distance does not seem to be in trade off with the other two soft constraints, but the other two soft constraints are. The results are quite time consuming to produce, so it has only been examined on this one instance.

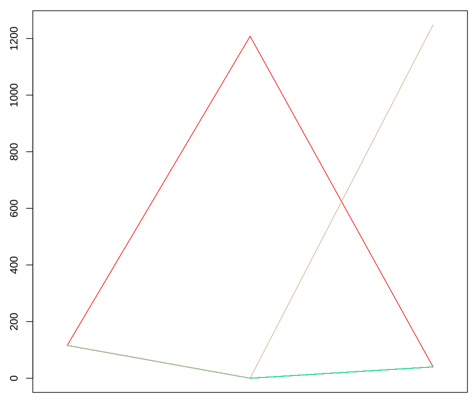
2.2 Heuristics

Here we will give a short description of the heuristic approach we implemented, in case the integer programming approach was not able to solve the problem. We also present some first results from using simulated annealing on the problem. Since we in parallel with working on meta heuristics solved the instances to optimality with integer programming, the analysis here is not very thorough.

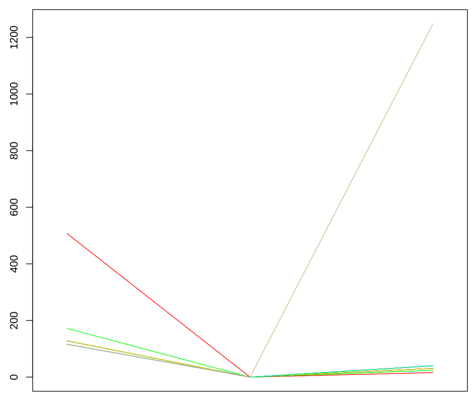
We represent the problem as a graph coloring problem. We have a node in the graph for each lecture and a precolored node for each timeslot. Restrictions are modelled by introducing edges between lecture-nodes or edges between lectures nodes and precolored timeslot-nodes.



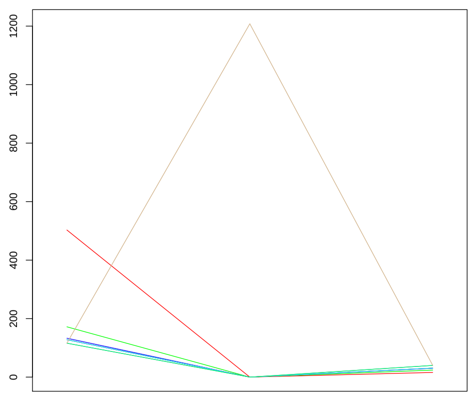
(a) All results in one graph



(b) Soft cost 1 (A) fixed



(c) Soft cost 2 (B) fixed



(d) Soft cost 1 (C) fixed

Figure 1: Four different parallel coordinate plots.

We try to solve the problem by a three phase approach. In the first phase we use a construction heuristic to give a starting solution. It colors nodes starting with the most restricted to the least restricted. If a node cannot be given a valid color, it is given the first color (color 0). In the second phase we use simulated annealing to find a valid solution. In the last phase we again use simulated annealing, but this time we try to minimize the soft cost, while making sure the solution is always valid. We only switch from phase two to phase three when a valid solution have been found. In the second phase we also only perform moves that either keep the same distance to feasibility or lower it.

Two neighborhoods have been implemented, the 1-move neighborhood and the 2-move neighborhood. When performing a 1-move a node the in graph is colored with another color, corresponding to moving a lecture from one timeslot to another. When a 2-move is done two nodes in graph swap colors, corresponding to exchanging two lectures.

The annealing process is implemented by a constant cooling factor which lowers the temperature after each move. When the temperature gets low enough (0.1) the cooling process is restarted by reseting the temperature. Before every restart the time is checked and if we have used more than 5 minutes the process is stopped, and the best solution found is used. This gives running times between 5 and 6 minutes.

Since recalculating the soft cost or number of hard conflicts every time we need to evaluate a move is very expensive, we instead store the already know soft cost value or the number of hard conflicts and update this. A further improvement could be implemented by also doing this for the distance to feasibility.

The results seen in table 2.2 have all been calculated with seed 1.

Instance	1-move	1 and 2-move	Optimal
E04	86	95	84
E05	83	107	83
E06	15	27	11
F05	167	68	44
F06	79	82	79
comp-2007-2-15	180	424	156
comp-2007-2-16	241	587	197
comp-2007-2-7	420	676	372
comp-2007-2-8	333	615	280

Table 2: Results from simulated annealing

For all instances except F05 the 1-move version performs far better than the 1 and 2-move version. The 1-move version does not perform well on the instance F05, because it is not possible to perform any 1-moves (all timeslot are used by a lecture). For two of the IMADA-instances the 1-move version finds the optimal solution. These first results suggest that one should use the 1-move version, and only if the results there seem bad the 2-move version should be tried.

For the small IMADA-instances then solutions found with the 1-move version are close to the optimal ones (when ignoring F05). For the large instances the results from using simulated annealing are significantly worse that the optimal ones. These results point in the direction that using simulated annealing on large timetabeling problems is not a good idea, however a more complete test should be done.

3 Extensions

3.1 Minimum number of rooms

Here we consider the basic problem and try to find the minimum number of rooms for which there still exits a feasible solution. For each instance we have lowered the number of available rooms by one each step, and the soft cost results are shown in the tables below. For each test run, the contribution from each of the three soft costs are shown. We continued to decrease the number of rooms until a feasible schedule no longer existed.

In table 3.1 the results for the large instance comp-2007-2-15 can be seen. When lowering the available rooms from 10 to 9 or 8 there is no change in the quality of the solution. When there are only 7 or 6 rooms available there is a small increase in the soft cost contribution from students having to many lectures in one day. For 5 and 4 rooms there is a large increase in the contribution from lectures in the first and last time slot, and a small decrease in the number of students with four or more lectures in a day. For 3 rooms it is not possible to find a feasible solution. In table 3.1 similar results can be seen for the instance comp-2007-2-8.

The results seem intuitive, since when the number of rooms decrease some lectures can no longer be done in parallel and must be moved to another time slot. If the lecture is moved to the first or last time slot of a day, the soft cost increase is the number of students in the course. If it is possible to find a time slot that is not the first or last of a day, the soft cost increase might just be a few students who now have four lectures or more on that day.

Rooms	Cost	First and last	Num. of Lectures	Separation
9	(0, 156)	116	40	0
8	(0, 156)	116	40	0
7	(0, 157)	116	41	0
6	(0, 159)	116	43	0
5	(0, 188)	152	36	0
4	(0, 324)	288	36	0
3	(230, 352)	316	36	0

Table 3: comp-2007-2-15: Soft cost results for finding the minimum number of rooms

3.2 An extra room

We have also considered increasing the number of rooms by one, to see if this gives a better solution. For all the large instances the is no gain from using 11 or 21 rooms instead of 10 or 20. The solution have exactly the same quality. From the tests in section 3.1 this is expected since the same quality solution could be found with fewer rooms.

For all the IMADA-instances a better solution is found by increasing the number of rooms to two. For all instances the soft cost contribution from lectures in the first or last timeslot of a day is lowered. The soft cost contribution from four or more lectures in a day stays almost the same. For the two spring schedules there is a change in the cost contribution from

Rooms	Cost	First and last	Num. of Lectures	Separation
19	(0, 280)	249	31	0
18	(0, 280)	249	31	0
...	...	...	...	...
7	(0, 280)	249	31	0
6	(0, 307)	275	32	0
5	(0, 435)	412	23	0
4	(18, 620)	591	29	0

Table 4: comp-2007-2-8: Soft cost results for finding the minimum number of rooms

	Original		Extra room	
Instance	Soft cost	Composition	Soft cost	Composition
E04	84	(78,6,0)	21	(19,2,0)
E05	83	(59,24,0)	66	(42,24,0)
E06	11	(11,0,0)	0	(0,0,0)
F05	44	(42,2,0)	32	(17,2,13)
F06	79	(46,8,25)	43	(17,9,17)

Table 5: Comparison of soft cost for original IMADA-instances with soft cost for an extra room

separation of lectures. The largest improvements clearly come from reducing the contribution from early or late lectures.

The fact that the improvement comes from early or late lectures is not suprising. The IMADA-schedules, with one room, all make heavy use of the first or last timeslot. Adding a room gives possibilities to reschedule these lectures to the middle timeslots. Rescheduling a lecture reduces the soft cost for all students in the course. For the two other soft cost contributions the gains are either smaller or more difficult to get. Rescheduling a lecture might reduce the number of lectures in a day to less than four, but most often only for a few students. Rescheduling a lecture so that there are two days between the lectures in a course is difficult, since two or three days cannot be used due to the already scheduled lecture in that course.

3.3 Rescheduling mandatory courses

If we are allowed to reschedule mandatory courses as well as elective courses, the new optimal solution will obviously be no worse than the optimal solution when we are only allowed to schedule elective courses.

When we tried to solve this new problem with integer programming, we were only able to solve the the IMADA-instance E04. All other instances took too long. The soft cost for the solved IMADA-instance was 0.

3.4 How many courses can a student take?

The thing we consider is, if it is possible to give a guideline to students on how many mandatory and elective courses they can take, while it is still very likely that we can find a valid schedule.

To investigate this problem we randomly generated students with four course enrollments. We ran five different tests, one for each of the following combinations of mandatory and elective courses: (4,0), (3,1), (2,2), (1,3) and (0,4). The first situation (when a student takes four mandatory courses and zero elective) is trivial since we do not have to schedule anything. The number of randomly genrerated students was chosen so that the total number of course enrollments matched the number of enrollments in the instance. Meaning that when running a test on IMADA-instance F05 we had about $\frac{195}{4}$ students, so that the total number of enrollments matched the 195 from the original instance.

We ran the tests on the two instances F05 and E04. Each test was run 10 times and the results can be seen in table 3.4 and 3.4.

Mandatory-Elective	Num. not valid	Num. feasible	Avg. quality
4-0	0	10	(0,0)
3-1	1	0	(13.2,74.8)
2-2	6	0	(29.0,140.8)
1-3	2	0	(31.3,142.6)
0-4	0	10	(0,122.6)

Table 6: Schedule quality for randomized students (instance F05)

Mandatory-Elective	Num. not valid	Num. feasible	Avg. quality
4-0	0	10	(0,0)
3-1	0	7	(2.6,41.6)
2-2	2	1	(31.9,83.5)
1-3	0	8	(5.8,140.5)
0-4	0	10	(0,18.6)

Table 7: Schedule quality for randomized students (instance E04)

From the results we see that the most difficult situation is when the students take two mandatory and two elective courses. Besides from this situation it seems that a valid schedule can quite often be found.

When also considering that in the tested situations students takes completely random courses, in contrary to the situation at IMADA where students only take some combinations of mandatory courses (the ones defined in their recommended study plan) and for a large part only one type of elective courses (computer science courses or math courses). This artificial situation of completely random course enrollments ,we believe, is a lot more difficult, than real life data would be. We consider it therefor likely that a valid schedule exists when considering students who takes no more than four courses. For more than two mandatory and two elective courses, we only seldomly found valid schedules.