Optimal Student Sectioning at Niederrhein University of Applied Sciences

Steffen Goebbels, Timo Pfeiffer
Faculty of Electrical Engineering and Computer Science,
Niederrhein University of Applied Sciences,
Reinarzstr. 49, 47805 Krefeld, Germany, +49-2151-822-4633
Steffen.Goebbels@hsnr.de, Timo.Pfeiffer@stud.hn.de

This is a self-archived pre-print version of a conference paper
that will appear in the proceedings of Operations Research 2019
conference (OR 2019 Dresden), Springer, Berlin

Abstract

Degree programs with a largely fixed timetable require centralized planning of student groups (sections). Typically, group sizes for exercises and practicals are small, and different groups are taught at the same time. To avoid late or weekend sessions, exercises and practicals of the same or of different subjects can be scheduled concurrently, and the duration of lessons can vary. By means of an integer linear program, an optimal group division is carried out. To this end, groups have to be assigned to time slots and students have to be divided into groups such that they do not have conflicting appointments. The optimization goal is to create homogeneous group sizes.

1 Introduction

A large number of articles deals with the "University Course Timetable Problem", see [1, 5] and the literature cited there. Here we are concerned with the subproblem "student sectioning", more precisely with "batch sectioning after a complete timetable is developed", see [3, 4, 6] for a theoretical discussion and overview. At our faculty for electrical engineering and computer science, we perform student sectioning on a fixed time table that already provides general time slots for groups, see Figure 1. Based on enrollment data, also the number of groups per lecture, exercise and practical is known. The setting is typical for technical courses at universities of applied sciences. Groups for a given subject might be taught weekly or only in every second or every fourth week. This gives freedom to choose the starting week for such groups and to place up to four groups in the same time slot. These groups can be taught in alternating weeks. Based on the time table's general time slots, our student sectioning problem has to select suitable slots and suitable starting weeks for groups. It also has to assign students to groups such that groups become nearly equal in size for each subject. A somewhat similar model with a different optimization goal is presented in [7]. For example, the often cited technique presented in [2] does not provide automated assignment of groups to time table slots. In the next section, we describe our model. Using IBM's solver ILOG CPLEX 12.8.0[1], we applied the model with real enrollment data to our time table. Section 3 summarizes results.

2 Model

For simple terminology, lectures, exercises and practicals that require group division are considered to be separate modules which are numbered by $1,\dots,N$.

Every module $k \in [N] := \{1,\dots,n\}$ is taught for n_k groups $G_{k,j}$, $j \in [n_k]$. The number of groups is determined in advance, based on current enrollment figures and teaching capacities. Each module k can be offered

[1] see https://www.ibm.com/customer-engagement/commerce

	8:00 9:00	9:00 10:00	10:00 11:00	11:00 12:00	12:00 13:00	13:00 14:00	14:00 15:00	15:00 16:00	16:00 17:00	17:00 18:00	18:00 19:00	19:00 20:00	20:00 21:00
Monday			*ENG F* Dj		*BSY V* Po		*MAR V* Gp		*MAR U* Gp				
	OOA P St								*OOA T* ShI				
	OOA P Dv												
	BSY P Ni												
Tuesday	*OOA P* Gb				*ALD V* Ue		*ALD U* Ue		*MAR U* Gp				
	BSY P Ne						*TEI2 U* Na		*BSY U* Po				
	BSY P Po												
	TEI2 P Na						*BSY U* Po			*MINT T* ShE			
	OOA P Qi												
Wednesday	*OOA U* St		*MA2 V* Tp		*OOA V* St		*OOA U* St		*ALD U* Ue				
	MA2 U Tp				*OOA V* Dv		*ALD U* Ue		*OOA U* Dv				
	ENG F Nn												
Thursday	*TEI2 V* Ha		*OOA V* St		*MA2 T* ShI2		*ENG F* Ge		*OOA U* Dv		*MINT T* ShI		
			OOA V Dv				*MA2 U* Su		*MA2 U* Su				
							ALD U Ue						
							ENG F Db						
Friday	*MA2 V* Tp		*MA2 U* Tp		*TEI2 V* Ha		*TEI2 P* Ha						
			TEI2 U Ha				*OOA P* Le						

Figure 1: Schedule of the bachelor course in computer science, second semester: the first acronym refers to the subject, then P denotes practical, U stands for exercise, V for lecture, T for tutorial and F for language course. Practicals, exercises (including language courses) and parallel lectures are subject to group planning. The last acronym denotes the lecturer. In this plan, all practicals but "OOA P" have a four-week frequency. Groups for "OOA P" are taught every second week. Exercises "MA2 U", "ALD U", and "OOA U" have a weekly frequency, the other exercises are taught every second week. Lecture "OOA V" is split into two groups.

on at most m_k time slots $T_{k,1},\dots,T_{k,m_k}$ per week, cf. Figure 2. Not all time slots might be needed, for example if $n_k < m_k$. For the participants of each group, a module takes place either weekly ($p_k := 1$), every second week ($p_k := 2$) or every fourth week ($p_k := 4$), see Table 1. If a module k is given in every second week, then at $T_{k,i}$ two groups can be planned in alternating weeks. This allows us to split $T_{k,i}$ into two simultaneous sub-slots $T_{k,i,1}$ and $T_{k,i,2}$. At most one group can be assigned to each of these sub-slots. The third index indicates whether the module is given for the assigned group in odd or even weeks. In the other week, participating students can take part in another bi-weekly module. If a module is given weekly, we only use a sub-slot $T_{k,i,1}$, for modules given every fourth week, time sub-slots $T_{k,i,l}$, $l \in [4]$, are considered. However, due to the workload of the instructors, there may be restrictions for assigning groups to time slots. The variables $1 \le q_{k,i} \le p_k$, $\sum_{i=1}^{m_k} q_{k,i} \ge n_k$, indicate the maximum number of groups that can be assigned to slots $T_{k,i}$, i.e., the maximum number of sub-slots with group assignment. For each group $G_{k,j}$ we de-

Table 1: Left: a group assigned to a sub time-slot $T_{k,i,l}$ is taught depending on frequency p_k in alternating weeks. Right: A student can be member of groups of different modules that are taught in overlapping time slots but in different weeks ($p_{k_1} := 2$, $p_{k_2} =: 4$, and $p_{k_4} =: 4$)

frequency p_k	week 1	week 2	week 3	week 4
1	$T_{k,i,1}$	$T_{k,i,1}$	$T_{k,i,1}$	$T_{k,i,1}$
2	$T_{k,i,1}$	$T_{k,i,2}$	$T_{k,i,1}$	$T_{k,i,2}$
4	$T_{k,i,1}$	$T_{k,i,2}$	$T_{k,i,3}$	$T_{k,i,4}$

week 1	week 2	week 3	week 4
$T_{k_1,i_1,1}$		$T_{k_1,i_1,1}$	
	$T_{k_2,i_2,2}$		
			$T_{k_3,i_3,4}$

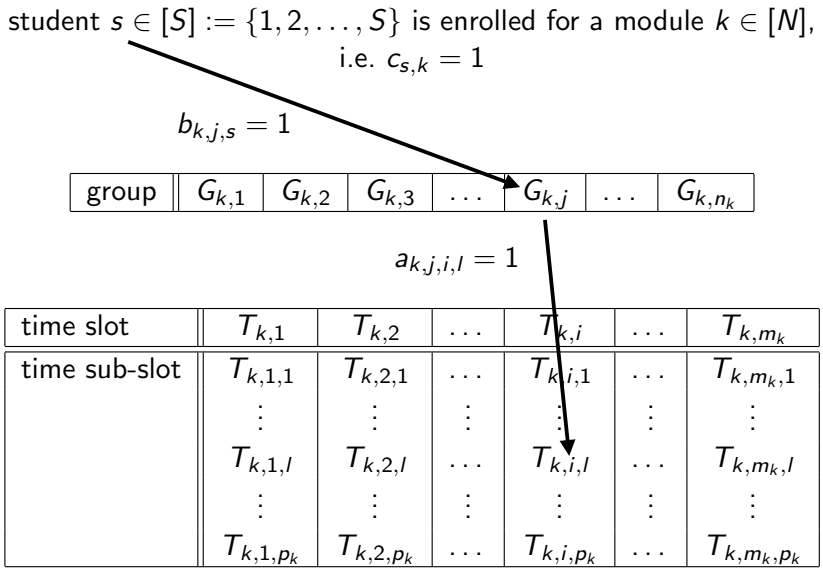

Figure 2: Model and notation

fine binary variables that assign a time sub-slot to the group. Let $a_{k,j,1,1}, \ldots, a_{k,j,1,p_k}$, $a_{k,j,2,1}, \ldots, a_{k,j,2,p_k}, \ldots$, $a_{k,j,m_k,1}, \ldots, a_{k,j,m_k,p_k} \in \{0,1\}$ with

$$\sum_{i=1}^{m_k} \sum_{l=1}^{p_k} a_{k,j,i,l} = 1. \tag{1}$$

If $a_{k,j,i,l} = 1$, the lesson for group $G_{k,j}$ takes place on time sub-slot $T_{k,i,l}$. Every sub-slot $T_{k,i,l}$ has to be assigned at most once ($k \in [N]$, $i \in [m_k]$, $l \in [p_k]$):

$$\sum_{j=1}^{n_k} a_{k,j,i,l} \leq 1. \tag{2}$$

Only $q_{k,i}$ groups may be scheduled for a time slot $T_{k,i}$, i.e., for all $k \in [N]$, $i \in [m_k]$:

$$\sum_{j=1}^{n_k} \sum_{l=1}^{p_k} a_{k,j,i,l} \leq q_{k,i}. \tag{3}$$

Let $S \in \mathbb{N}$ be the number of all students. Each student can register for the modules individually or, as in the case of English courses, is automatically assigned based on her or his level. Students can select modules that belong to different semesters, as modules may have to be repeated. We use a matrix $C \in \{0,1\}^{S \times N}$ to describe whether a student has enrolled for a module. Thereby, $c_{s,k} = 1$ indicates that student s has chosen module k. We have to assign exactly one group to each student $s \in [S]$ for each chosen module. To this end, we use binary variables $b_{k,j,s} \in \{0,1\}$. Student s is in group j of module k iff $b_{k,j,s} = 1$. For $k \in [N]$ and $s \in [S]$, we get condition

$$\sum_{j=1}^{n_k} b_{k,j,s} = c_{s,k}. \tag{4}$$

An external group assignment takes place for some modules (language courses). In this case, variables $b_{k,j,s}$ have to be set accordingly.

However, there must be no collision with simultaneous group assignments, cf. Table 1. Each time slot consists of one to four hours in a fixed time grid covering one week. Per module, the duration of time slots is (approximately) the same. Each week can be modeled with the set $[50]$ representing hours, i.e., $T_{k,i}$, $T_{k,i,l} \subset [50]$. It is allowed that the hours of time slots T_{k,i_1} and T_{k,i_2} (of the same module k) overlap, i.e. $T_{k,i_1} \cap T_{k,i_2} \neq \emptyset$ for $i_1 \neq i_2$, only if the module is given simultaneously by several instructors in different rooms.

- If a student is in a weekly group of one module, he may not be in a time-overlapping group of another module.
- If a student is in a bi-weekly group on a time sub-slot with a third index l, he may not be in another bi-weekly group on a time-overlapping sub-slot with the same third index l. He also must not be assigned to a group that belongs to an overlapping four-weekly time sub-slot with a third index l or $l+2$.

- If a student belongs to a group that is placed on a four-weekly time sub-slot with a third index l, he must not be in another group belonging to an overlapping four-weekly time sub-slot with the same third index l.

Conflicting time sub-slots are calculated in advance. Let T_{k_1,i_1,l_1} and T_{k_2,i_2,l_2}, $k_1 \neq k_2$, be two conflicting time slots for which, according to previous rules, two non-disjoint groups cannot be assigned. Group G_{k_1,j_1} is assigned to time sub-slot T_{k_1,i_1,l_1} iff $a_{k_1,j_1,i_1,l_1} = 1$, and a student s is assigned the group G_{k_1,j_1} iff $b_{k_1,j_1,s} = 1$. If also group G_{k_2,j_2} is assigned to time sub-slot T_{k_2,i_2,l_2} via $a_{k_2,j_2,i_2,l_2} = 1$ and if student s is assigned to this group via $b_{k_2,j_2,s} = 1$, then there is a collision. Thus,

$$a_{k_1,j_1,i_1,l_1} + b_{k_1,j_1,s} + a_{k_2,j_2,i_2,l_2} + b_{k_2,j_2,s} \leq 3 \quad (5)$$

has to be fulfilled for all colliding pairs $(T_{k_1,i_1,l_1}, T_{k_2,i_2,l_2})$ of time sub-slots, all groups $j_1 \in [n_{k_1}]$, $j_2 \in [n_{k_2}]$ and all students $s \in [S]$.

Collisions between group assignments are not defined independently of students by rule (5). This leads to a significant combinatorial complexity that has to be reduced. To speed-up the algorithm, certain groups can be assigned to sub-slots in a fixed manner. This can be done easily if, for a subject k, the number of sub-slots $m_k \cdot p_k$ equals the number of groups n_k. For such modules k we can set

$$a_{k,j,i,l} := \begin{cases} 1 & : \quad j = (i-1) \cdot p_k + l \\ 0 & : \quad \text{otherwise.} \end{cases} \quad (6)$$

By assigning groups to sub-slots in a chronologically sorted manner due to their group number, one can also avoid many permutations. Sorting can be established by following restrictions for all modules $k \in [N]$, all time slots $i_1 \in [m_k]$ and all sub-slots T_{k,i_1,l_1}, $l_1 \in [p_k]$, and all groups $j_1 \in [n_k]$:

$$\max \left\{ \sum_{i_2=i_1+1}^{m_k} \sum_{j_2=1}^{j_1-1} \sum_{l_2=1}^{p_k} a_{k,j_2,i_2,l_2}, \sum_{j_2=1}^{j_1-1} \sum_{l_2=l_1+1}^{p_k} a_{k,j_2,i_1,l_2} \right\} \leq n_k \cdot (1 - a_{k,j_1,i_1,l_1}). \quad (7)$$

The inequality can be interpreted as follows. If group j_1 has been assigned to sub-slot T_{k,i_1,l_1} then no group with smaller index $j_2 < j_1$ must be assigned to a "later" sub-slot T_{k,i_2,l_2} in the sense that either $i_2 \geq i_1$ or $i_2 = i_1$ and $l_2 > l_1$.

Dual education and part-time students s may only be divided into those groups of their semester, that are assigned to time slots on certain days. This restriction does not apply to modules that do not belong to the students' semester (repetition of courses). For all time sub-slots $T_{k,i,l}$, at which s cannot participate, we require for $j \in [n_k]$:

$$a_{k,j,i,l} + b_{k,j,s} \leq 1. \quad (8)$$

Two (but no more) students s_1 and s_2 can choose to learn together. Then they have to be placed into the same groups of the modules that they have both chosen. This leads to constraints if $c_{s_1,k} = c_{s_2,k}$, $k \in [N]$. Then for all $j \in [n_k]$

$$b_{k,j,s_2} = b_{k,j,s_1}. \quad (9)$$

Students should be assigned to groups such that, for each module, groups should be of (nearly) equal size (cf. [2]). To implement this target, we represent the difference of sizes of groups j_1 and j_2 of module k with the difference of two non-negative variables Δ^+_{k,j_1,j_2}, $\Delta^-_{k,j_1,j_2} \geq 0$:

$$\Delta^+_{k,j_1,j_2} - \Delta^-_{k,j_1,j_2} = \sum_{s=1}^{S} (b_{k,j_1,s} - b_{k,j_2,s}). \quad (10)$$

Thus, under constraints (1)–(5) and (8)–(10) we have to minimize following objective function:

$$\sum_{k=1}^{N} \sum_{j_1=1}^{n_k-1} \sum_{j_2=j_1+1}^{n_k} (\Delta^+_{k,j_1,j_2} + \Delta^-_{k,j_1,j_2} - \varepsilon_{k,j_1,j_2}). \quad (11)$$

We observed long running times if an uneven number of students have to be divided into an even number of groups, and vice versa. To further simplify complexity, we propose to subtract float or integer variables $0 \leq \varepsilon_{k,j_1,j_2} \leq \min\{D, \Delta^+_{k,j_1,j_2} + \Delta^-_{k,j_1,j_2}\}$ within the objective function (11). They serve as slack variables that allow absolute group size differences to vary between 0 and $D \in \mathbb{N}_0 = \{0,1,2,\dots\}$ without penalty. Significant speedup already is obtained for $D = 1$, see Section 3. Thus, we consider a group sectioning as being optimal even if there exist slightly better solutions with fewer differences.

In certain groups j, a contingent of r places can be reserved (e.g., for participants of other faculties). This is done by adding or subtracting the number r on the right side of (10): If $j = j_1$, then r has to be added, if $j = j_2$, then r is subtracted.

Table 2: CPLEX 12.8.0 processor times measured in seconds on an Intel Core i5-6500 CPU, 3.20 GHz x4 with 16 GB RAM

slack size D	sorting (7)	initialization (6)	running time computer science	running time electrical engineering
2	-	-	51.59	0.13
2	-	✓	3.35	0.1
2	✓	-	2.8	0.06
2	✓	✓	1.57	0.05
1	-	-	57.92	0.15
1	-	✓	6.32	0.08
1	✓	-	3.49	0.09
1	✓	✓	3.33	0.07
0	- / ✓	- / ✓	memory overflow	≤ 2.65

3 Results

We merged planning and enrollment data from different sources to generate the input for the integer linear program. It can be applied separately for each field of study. Presented results belong to our bachelor programs in computer science (second and fourth semester, 330 students including 59 dual education and part-time students, 30 modules, up to 8 groups per module) and electrical engineering (second, fourth, and sixth semester, 168 students including 40 dual education and part-time students, 27 modules, up to 4 groups per module). Table 2 summarizes running times with respect to combinations of speed-up measures $D \in \{1,2\}$, sorting (7), and fixed assignment of certain groups to time-slots (6). Choosing $D = 0$ leads to memory overflow after eight hours in case of computer science (independent of speed-up measures), whereas group division for electrical engineering finishes in 2 65 seconds (without speed-up measures).

4 Enhancements

Some students miss the date for enrollment. To assign them to groups, one can also use the integer linear program as an online algorithm. Then one adds students to an existing group sectioning by maintaining all previously done assignments.

If students choose modules from different semesters then the existence of a feasible solution is not guaranteed. However, such situations could be identified prior to group planning. Alternatively, one can deal with such students by applying the online version of the algorithm in order to individually identify conflicts.

As a secondary optimization goal, one could maximize the number of students that get commonly assigned to groups along all modules. Students who have to repeat modules could be distributed as evenly as possible among groups, since experience has shown that for such students the risk of non-appearance is high.

References

[1] Bettinelli, A., Cacchiani, V., Roberti, R., Toth, P.: An overview of curriculum-based course timetabling. TOP **23**(2), 313–349 (2015)

[2] Laporte, G., Desroches, S.: The problem of assigning students to course sections in a large engineering school. Computers & Operations Research **13**(4), 387 – 394 (1986)

[3] Müller, T., Murray, K.: Comprehensive approach to student sectioning. Annals of Operations Research **181**(1), 249–269 (2010)

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

[5] Schimmelpfeng, K., Helber, S.: Application of a real-world university-course timetabling model solved by integer programming. OR Spectrum **29**(4), 783–803 (2007)

[6] Schindl, D.: Student sectioning for minimizing potential conflicts on multi-section courses. In: Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling (PATAT 2016), pp. 327–337. Udine (2016)

[7] Sherali, H.D., Driscoll, P.J.: Course scheduling and timetabling at USMA. Military Operations Research **4**(2), pp. 25–43 (1999)