Timetabling Problems at the TU Eindhoven

John van den Broek, Cor Hurkens, and Gerhard Woeginger

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, gwoegi}@win.tue.nl

Abstract. The students of the Industrial Design department at the TU Eindhoven are allowed to design part of their curriculum by selecting courses from a huge course pool. They do this by handing in ordered preference lists with their favorite courses for the forthcoming time period. Based on these informations (and on many other constraints), the department then assigns courses to students. Until recently, the assignment was computed by human schedulers who used a quite straightforward greedy approach. In 2005, however, the number of students increased substantially, and as a consequence the greedy approach did not yield acceptable results anymore.
This paper discusses the solution of this real-world timetabling problem: We present a complete mathematical formulation of it, and we explain all the constraints resulting from the situation in Eindhoven. We present an elegant integer linear programming model for this problem that easily can be put into CPLEX. Finally, we report on our computational experiments and results around the Eindhoven real-world data.

Keywords: University timetabling; network flow formulation; NP-completeness; integer programming formulation.

1 Introduction

In February 2005, outraged students of the Industrial Design department were protesting at the TU Eindhoven (The Netherlands). Uproar and revolt were in the air. What had happened? Here is the story. The academic year of these roughly 350 students of Industrial Design is split into a number of periods. In every period, every student must do a number of small *courses*. There is a pool of roughly 55 courses to choose from, and every student submits an ordered preference list with his/her 10 favorite courses to the department. Based on all the ordered preference lists, a *scheduler* at the department then assigns roughly 4 courses to every student. Until recently, the scheduler was a human decision-maker who essentially applied a hand-woven greedy assignment procedure.

In February 2005, the students were absolutely dissatisfied with the work of the human scheduler: Many of them did not get the courses which they would have liked to get. Many of them were assigned to courses which they really did not want to do. And more than 150 out of the 350 students received courses that were not listed on their preference list!

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 141–156. ISBN 80-210-3726-1.

The department of Industrial Design realized that they had a problem. They also realized that they did not know how to settle this problem. The number of students had increased substantially, and the timetabling problem had become much larger, much harder, and much more complex. Hence, the department contacted the local experts on the campus: Us. They were hoping to find a somewhat better assignment through computer programs. They explained their problem to us (in a certain problem formulation No. 1), and we happily told them that we are able to solve it: The problem (in formulation No. 1) could be modeled as a network flow problem, and hence is solvable in polynomial time. Unfortunately, it turned out that formulation No. 1 was not a complete formulation of the problem: They had forgotten to inform us about a number of additional restrictions that lead to a new, more difficult assignment problem (in formulation No. 2), which eventually turned out to be NP-hard.

This paper is a report on the course assignment problem of the Industrial Design department: We will describe the assignment problem in its (incomplete) formulation No. 1 and in its (complete) formulation No. 2. We show that formulation No. 1 yields a tractable problem, whereas formulation No. 2 yields an intractable problem. Our main contribution is a careful case study of the complete problem formulation. We design an elegant integer linear programming model for it, with roughly 9000 variables and roughly 7000 constraints. Putting this ILP model into CPLEX yields excellent results within moderate computation times. We describe the ILP model in detail, and we report on our computational experiments with the real-world data of the Industrial Design department.

Structure of the paper. The rest of the paper is structured in the following way. In Section 2 we give a literature review of university and school time tabling. Section 3 contains a detailed description of the problem we solved for the department of Industrial Design. The problem is formulated as an integer linear program which will be described in Section 4. Section 5 contains the computational results. Some conclusions are given in Section 6.

2 Literature Review

The literature contains a large number of variants of the timetabling problem. These variants differ from each other by the type of institution involved (university or high school) and by the type of constraints. The annotated bibliography of timetable construction by Schmidt & Ströhlein [12] lists many papers that appeared before 1980. Schaerf [11] gives a survey of the various formulations of timetabling problems and classifies the timetabling problem into the following three main classes:

School timetabling: The weekly scheduling of all the classes of a high school. Avoid that teachers meet two classes at the same time, and avoid that classes meet two teachers at the same time.

Examination timetabling: The scheduling of the exams of several university courses. Avoid that exams of courses with common students overlap. Spread out the exams for every student as much as possible over time.

Course timetabling: The weekly scheduling for all the lectures of several university courses. Minimize the overlaps of lectures of courses with common students.

Of course this classification is crude, and there are many real-world timetabling problems that fall in between two of these classes.

The basic school timetabling problem is also known as the *class-teacher model*. The simplest problem consists in assigning lectures to periods in such a way that no teacher or class is involved in more than one lecture at a time. Even, Itai & Shamir [5] proved that there always exists a solution of this simplest version of the school timetabling problem, unless a teacher or class is involved in more lectures than there are time slots. Alternative formulations of the school timetabling problem with more constraints can be found for example in Even, Itai & Shamir [5], Garey & Johnson [7] and de Werra [4].

The main differences between course timetabling and examination timetabling are that examination timetabling has only one exam for each course, that the time conflict condition is strict, and that several exams can be done simultaneously in one room. Examples for additional soft constraints are: Students can do at most one exam per day, and students may not have too many consecutive exams. Schaerf [11] gives an integer linear programming formulation of the examination timetabling problem and describes some alternative variants of the problem.

The course timetabling problem consists in scheduling a set of lectures for each course within a given number of rooms and time period. The main difference from the school timetabling problem is that university courses can have common students, whereas school classes are disjoint sets of students. De Werra [4] gives a binary integer programming formulation. Schaerf [11] discusses some of the most common variants of the basic formulation.

One variant is called the *grouping subproblem* or *student scheduling problem*. If the number of students is too large for one room, courses are split into groups of students and there are conditions on the minimum and maximum number of students that can be assigned to each group. A student is required to take a certain number of courses, which they have to select themselves after a timetable is made available. The problem consists of assigning a student to a specific group of a course for a given fixed timetable such that students are satisfied and there are no time conflicts, see Busam [2], Feldman & Golumbic [6] and Laporte & Desrochers [8].

Cheng, Kruk & Lipman [3] discuss the Student Scheduling Problem (SSP) as it generally applies to high schools in North America. They define the problem as the assignation of courses and a specific section to each student. The objective is to fulfil student requests, providing a conflict-free schedule. They show that the problem is NP-hard and discuss various multi-commodity flow formulations with fractional capacities and integral gains. The main difference between the SSP and our timetabling problem is that for the SSP all courses on the preference list of the students have to be assigned to students. This results in most practical cases into an empty feasible solution set.

Laporte & Desrochers [8] give a mathematical formulation of the student scheduling problem. They formulate the problem as an optimization problem splitting the requirements into hard and soft ones. The only hard constraint in their model is that student course selections must be respected. Time conflicts for students are soft constraints. When time conflicts occur students are advised to make a different course selection. The problem is then solved in three phases: In the first one the algorithm searches for an admissible solution, in the second section enrollments are balanced and in the third the room capacities have to be respected. Tripathy [13] formulated the student scheduling problem as an integer linear programming problem and uses Lagrangian Relaxation to solve it. Sabin & Winter [10] use a greedy approach that is moderated by an intelligent ordering of the students. Miyaji, Ohno & Mine [9] apply goal programming.

3 Problem Description

At our first meeting, the Industrial Design department explained the problem to us in a certain problem formulation No. 1; see Subsection 3.1. This problem can be modeled as a network flow problem, and hence is solvable in polynomial time; see Ahuja, Magnanti & Orlin [1].

Unfortunately, we learnt after some time that formulation No. 1 was *not* a complete formulation of the problem. They actually had forgotten to tell us about a number of additional restrictions that lead us to a new, more difficult assignment problem formulation No. 2. Subsection 3.2 describes formulation No. 2.

3.1 Problem Formulation No. 1

At the first meeting with the Industrial Design department, they told us that every student hands in a preference list of at most 10 courses and requests a certain number of courses. The only constraints are that a student can not do two courses at the same time and there is a maximum number of students that can be assigned to a course. This subsection contains a more detailed description of problem formulation No. 1.

A set C of courses and for each course c an *upper bound* C_c^{max} on the number of students is given. This number depends on the preference of the teacher and the room capacity in which the course is given. For each course also the weekly meeting time is already assigned. This weekly meeting time always consists of two consecutive hours. Two such consecutive hours are defined as one *time slot*. The weekly meeting time of a course is chosen from a set T of disjoint time slots. $T(c)$ is defined as the time slot which is the weekly meeting time of course c. Hence, one of the constraints in the model is that courses c_i and c_j can not be assigned to one student if $T(c_i) = T(c_j)$.

We define S as the set of students. For each student s the requested number r_s of courses is given. P_s is defined as the set of positions on the preference list for which student s filled in a course. Most students have $P_s = \{1, \ldots, 10\}$. There are also students that hand in a smaller preference list. For instance, a

student almost finishing his bachelor and only one course left to do, which has to be a math course, hands in a preference list with only math courses. For a student s with only six courses on its preference list we have $P_s = \{1, \ldots, 6\}$. Table 1 gives a few examples of preference lists. Column Pi gives the encoded course name of the course on position i of the preference list. The parameter c_{sp} is introduced and is equal to c if course c is on position p of the preference list of student s.

Table 1. Example of preference lists

Student	r_s	P1	P2	P3	...	P10
s040202	4	DAC03	DA247	DA125	...	DA405
s040203	4	DA619	DA125	DA201	...	DA616
s040204	4	DA418	DA242	DA402	...	DA621

In summary: The input of problem formulation No. 1 consists of:

- a set T of time slots.
- a set C of courses; for every course $c \in C$ a time slot $T(c)$ and a maximum number C_c^{max} of participating students is given.
- a set S of students; for every student $s \in S$ a set P_s of filled positions of the preference list, a course c_{sp} for each position $p \in P_s$ and a requested number r_s of courses is given.

The goal is to assign as many courses to students as possible, while:

- the number of courses assigned to student s does not exceed the requested number r_s.
- courses assigned to a student are on its preference list.
- courses assigned to a student do not conflict in time.
- no course exceeds its maximum number of assigned students.

This problem can be modeled as a network flow problem. A description of this network flow model is given in Appendix A.

3.2 Problem Formulation No. 2

As we received the first data set from the Industrial Design department, we were very surprised: there suddenly were also *lower bounds* C_c^{min} on the number of students participating in course c. This yields the new constraint that a course either will not be given at all, or otherwise has at least C_c^{min} participating students. This new constraint can not be modeled as a flow-constraint, and hence the maximum flow model in Appendix A becomes obsolete. In fact, the new constraint makes the problem NP-hard; see Appendix B. After looking at the data more carefully and after conversations with the Industrial Design

department we noticed there were a lot more restrictions. The remainder of this subsection explains these extra restrictions and defines the problem into more detail.

An academic year is divided into a certain number of periods. The length of such a period depends on the number of periods in which the academic year is split. For instance, the academic year 2005-2006 is divided into six periods of five weeks. We define such a period as a *block*. The Industrial Design department wants us to schedule two blocks simultaneously. Therefore, set B is introduced as the set of blocks that have to be scheduled simultaneously.

In problem formulation No. 1 we assumed the *workload* of all courses was equal. However, there are courses with a workload of 40 hours and courses with a workload of 80 hours. This can not be modeled with a flow constraint. In the remainder of this paper a workload of 1 corresponds with a workload of 40 hours. In Appendix B we prove that having courses with a workload 1 and courses with a workload 2 makes the problem already NP-hard. For each course $c \in C$ and block $b \in B$ the parameter $w(c, b)$ is defined as the workload of course c in block b. Hence for a course c with a workload of 80 hours in block b we have $w(c, b) = 2$.

In problem formulation No. 1, r_s was defined as the requested number of courses of student s. This definition is adjusted in problem formulation No. 2 into the requested workload of student s for $|B|$ blocks together. For every student s, a maximum requested workload r_{sb} for each block $b \in B$ separately is given, because the requested workload of a student is not always equally divided over all blocks $b \in B$. For instance, if blocks b_1 and b_2 have to be scheduled simultaneously and $r_s = 3$, then parameters r_{sb_1} and r_{sb_2} are both equal to 2. In this case the model is allowed to choose the block in which student s is assigned two courses. Another example, if student s has to do a practical training in block b_2 he has: $r_s = 2$, $r_{sb_1} = 2$ and $r_{sb_2} = 0$.

It was assumed in problem formulation No. 1 that a course has one meeting every week, hence it has one time slot. But there are also courses which have two weekly meetings, hence have two time slots. If such a course is assigned to a student, the student has to be available at both time slots. If courses with two time slots are introduced into problem formulation No. 1, the problem can not be modeled as a network flow problem.

The set C of courses offered to the students contains courses with multiple sections, meaning that the course is repeated during the week. Table 2 contains course DA242 as an example. The time slots in the table are encoded. For example, code B1TM2 stands for the second part of Tuesday morning in block 1. The workloads of a course in block 1 and 2 are denoted with $wlb1$ and $wlb2$. The course DA242 has five sections which all have two time slots as meeting times. The first meeting is for all sections on the same time slot and the other is on a different time slot for each section. The first meeting is a class in one large lecture room and the second is a meeting where exercises have to be made in smaller groups.

We define I as the set of sections offered to the students. For every section $i \in I$ its course $c(i) \in C$ is given. In problem formulation No. 2 there are

no maximum and minimum number of students for a course like in problem formulation No. 1, but a minimum number C_i^{min} and a maximum number C_i^{max} of students for each section $i \in I$. The meeting times for each section $i \in I$ are given as the set of time slots $T(i) \subseteq T$. There are a few courses, for example literature studies, which are not assigned to a time slot and thus $T(i) = \emptyset$.

Table 2. Examples of courses

Course	Section	Time slots of meetings	wlb1	wlb2	Min	Max
DA242	DAG242-1	B1TM2, B1TA1	1	0	0	30
	DAG242-2	B1TM2, B1TA2	1	0	0	30
	DAG242-3	B1TM2, B1WA1	1	0	0	30
	DAG242-4	B1TM2, B1WA1	1	0	0	30
	DAG242-5	B1TM2, B1WA2	1	0	0	30
DA247	DAG247-1	B1WA2, B2WA2	1	1	5	15
	DAG247-2	B1WA2, B2WA2	1	1	5	15

Another constraint arises if students have specific needs, for instance when they almost finish their studies and only have one course left to pass. Then a course on the preference list of the student can be set to *urgent*. As long as the maximum number of students (all with an urgency) is not assigned to this course, the course has to be assigned to the student. A course which is urgent for one student has to be given. In this case, it doesn't matter whether the minimum number of students is reached or not. We define U as the set containing all combinations (s, p) for which course c_{sp} is urgent for student s.

A few courses have meeting times which are spread over two blocks. See for example course DA247 in Table 2. This course has two sections and a total workload of two which is equally spread over the two blocks. If a student is assigned to a section of this course in one block he needs to be assigned to the same section of this course in the next block. Hence, it is also possible that courses are given in two blocks which are not scheduled simultaneously. If this occurs, this implies there are students already preassigned to sections if the schedule of the second block is made. Therefore, we introduce the set F of *fixations* which contains combinations (s, p, i) for which section i of course c_{sp} is already assigned to student s.

In summary: the input of problem formulation No. 2 consists of:

- a set B of blocks that have to be scheduled simultaneously.
- a set T of time slots.
- a set C of courses; for every course c its workload $w(c, b)$ for each block b is given.
- a set S of students; for every student s a total requested workload r_s, a requested workload r_{sb} for each block separately, a set P_s of filled positions on the preference list and for each position $p \in P_s$ a course c_{sp} is given.

- a set I of sections; for every section i its course $c(i)$, a minimum C_i^{min} and maximum C_i^{max} number of students and a set of time slots $T(i) \subseteq T$ is given.
- a set U of combinations (s, p) for which course c_{sp} is urgent for student s.
- a set F of combinations (s, p, i) for which section i of course c_{sp} is already preassigned to student s.

Our main goal is to assign workload to students as much as possible, while:

- maintaining the number of students in a section below a maximum size prescribed.
- the total workload assigned to student s is less than or equal to r_s.
- the workload assigned to student s in block b is less than or equal to r_{sb}.
- sections assigned to a student do not conflict in time.
- students are only assigned to a section of a course on their preference list.
- students are only assigned to one section of a course.
- student s is assigned to section i if $(s, p, i) \in F$.

Soft constraints are for example spreading students over sections, a section needs to be assigned to at least a certain minimum number of students and student s has to be assigned to course c_{sp} if $(s, p) \in U$.

4 The Integer Linear Programming Model

To build a schedule which best fits the needs for the students, the problem is split into four subproblems which are formulated as an integer linear programming problem. These subproblems are solved sequentially, keeping the objective value of the foregoing subproblems the same. The goals of the four subproblems are:

1. Maximize the number of assigned courses with an urgency.
2. Minimize the shortage of students to reach the minimum number of students of a section. Because of urgencies, some sections must be taught, but don't have enough students with this course on their preference list. We assign as many students as possible to those sections.
3. Maximize the total assigned workload. We try to assign a workload r_s to every student s.
4. 'Optimize' the timetable. For example by assigning courses to students which rank high on their preference list.

All parameters are already introduced in Section 3. Left to define are the decision variables. These are defined as follows:

$$x_{sp} := \begin{cases} 1 \text{ if course } c_{sp} \text{ is assigned to student } s \\ 0 \text{ otherwise} \end{cases}$$

$$y_i := \begin{cases} 1 \text{ if section } i \text{ is assigned to one or more students} \\ 0 \text{ otherwise} \end{cases}$$

$$z_{spi} := \begin{cases} 1 \text{ if section } i \text{ of course } c_{sp} \text{ is assigned to student } s \\ 0 \text{ otherwise} \end{cases}$$

The following constraints have to be fulfilled in all four subproblems:

$$x_{sp} = \sum_{i \in I | c_{sp} = c(i)} z_{spi} \qquad \forall s \in S, \forall p \in P_s \qquad (1)$$

$$\sum_{p \in P_s} \sum_{i \in I | c_{sp} = c(i)} w(c_{sp}, b) z_{spi} \leq r_{sb} \qquad \forall s \in S, \forall b \in B \qquad (2)$$

$$\sum_{p \in P_s} \sum_{i \in I | c_{sp} = c(i)} \sum_{b \in B} w(c_{sp}, b) z_{spi} \leq r_s \quad \forall s \in S \qquad (3)$$

$$\sum_{s \in S} \sum_{p \in P_s, c_{sp} = c(i)} z_{spi} \leq C_i^{max} y_i \qquad \forall i \in I \qquad (4)$$

$$\sum_{p \in P_s} \sum_{i \in I | c_{sp} = c(i), t \in T(i)} z_{spi} \leq 1 \qquad \forall s \in S, \forall t \in T \qquad (5)$$

$$z_{spi} = 1 \qquad \forall s \in S, \forall p \in P_s, \forall i \in I | (s, p, i) \in F \qquad (6)$$

$$x_{sp} \in \{0, 1\} \qquad \forall s \in S, \forall p \in P_s \qquad (7)$$

$$y_i \in \{0, 1\} \qquad \forall i \in I \qquad (8)$$

$$z_{spi} \in \{0, 1\} \qquad \forall s \in S, \forall p \in P_s, \forall i \in I \qquad (9)$$

Constraint (1) takes care that at most one section of a course is assigned to a student. The workload assigned to a student has to be less than or equal to the requested workload each block separately and all blocks together. This is fulfilled by constraints (2) and (3). Constraint (4) enforces that the maximum number of students for a section is not exceeded and constraint (5) takes care that at each time slot only one section is assigned to each student. If $(s, p, i) \in F$ then section i of course c_{sp} has to be assigned to student s, which is fulfilled by constraint (6).

As explained above, the problem is split into four subproblems which are solved sequentially. The goal of the first subproblem is to maximize the number of assigned courses with an urgency. The constraint that a section needs to have more than a minimum number of students is not a restriction in this subproblem, because at least one section of a course must be given if there is a student with an urgency for this course. This first subproblem can be solved with the following ILP formulation:

$$U^{max} = \max \sum_{(s,p) \in U} x_{sp}$$

$$\text{(x,y,z) satisfy (1)-(9)}$$

The next step is to minimize the shortage of students to reach the minimum number of students of a section, keeping the maximum number of assigned courses with an urgency equal to U^{max}. There are sections that have to be given because they are assigned to students with an urgency for the corresponding course. Those sections are assigned to other students such that the minimum number of students for those sections is reached. The decision variable s_i is defined as the shortage of students for section i, i.e. the minimum number C_i^{min} of students subtracted with the number of students assigned to section i. The second subproblem minimizes the total shortage S^{min} of students. This results into the following ILP formulation:

$$S^{min} = \min \sum_{i \in I} s_i$$

$$\begin{aligned} \textstyle\sum_{(s,p)\in U} x_{sp} &= U^{max} \\ \textstyle\sum_{s\in S}\sum_{p\in P_s, c_{sp}=c(i)} z_{spi} + s_i &\geq C_i^{min} y_i \ \forall i \in I \\ s_i &\in \mathbb{Z}^+, \qquad \forall i \in I \\ \text{(x,y,z) satisfy (1)-(9)} & \end{aligned}$$

The third subproblem maximizes the total workload assigned to students with the restrictions that U^{max} and S^{min} keep their optimal values. This maximum workload is denoted by W^{max} and is determined by the following model:

$$W^{max} = \max \textstyle\sum_{s\in S}\sum_{p\in P_s}\sum_{b\in B} w(c_{sp}, b) x_{sp}$$

$$\begin{aligned} \textstyle\sum_{i\in I} s_i &= S^{min} \\ \textstyle\sum_{(s,p)\in U} x_{sp} &= U^{max} \\ \textstyle\sum_{s\in S}\sum_{p\in P_s, c_{sp}=c(i)} z_{spi} + s_i &\geq C_i^{min} y_i, \ \forall i \in I \\ s_i &\in \mathbb{Z}^+, \qquad \forall i \in I \\ \text{(x,y,z) satisfy (1)-(9)} & \end{aligned}$$

To 'optimize' the final timetable we assign courses as high as possible on the preference lists, spread the students as equally as possible over the sections of a course and discourage that one student gets a lot of courses which are on the bottom of his preference list. Therefore, the fourth subproblem is solved. The objective function is separated into three terms and has to be minimized under the restrictions that U^{max}, S^{min} and W^{max} keep their optimal values.

The term in the objective function to assign courses as high as possible on the preference lists is: $W_p \sum_{s\in S}\sum_{p\in P_s}\sum_{b\in B} w(c_{sp}, b)(82 - (10-p)^2) x_{sp}$. Assigning a course on top of a preference list, $p = 1$ for this course, adds a lot less to the objective function than assigning a course on the bottom of the list, $p = 10$ for this course. W_p is a weighting factor and also the workload is taken into account.

If a course has multiple sections, students have to be spread as equally as possible over the sections. Therefore, I_c^{max} is introduced as the number of students assigned to the section of course c with the most students assigned. Also the spread S_c of course c is introduced and is equal to the sum over all sections of the difference between I_c^{max} and the assigned number of students in each section. S_c is added to the objective function with a weighting factor W_s.

We also discourage that one student gets a lot of courses of his 7th up to 10th position of his preference list. A constraint is added to the model that enforces that every student gets at most one course from these positions, else a penalty W_e is paid for each 'extra' course from these positions. Therefore, the decision variable E_s is introduced for every student s. This variable is equal to the 'extra' number of courses assigned to student s which are on the 7th up to 10th position of his preference list.

This results into the final ILP formulation:

$$\min W_p \sum_{s\in S}\sum_{p\in P_s}\sum_{b\in B} w(c_{sp},b)(82-(10-p)^2)x_{sp} + W_s \sum_{c\in C} S_c + W_e \sum_{s\in S} E_s$$

$$\begin{aligned}
I^{max}_{c(i)} &\geq \textstyle\sum_{s\in S}\sum_{p\in P_s, c_{sp}=c(i)} z_{spi} && \forall i \in I\\
S_c &= \textstyle\sum_{i\in I|c=c(i)}(I^{max}_c - \sum_{s\in S}\sum_{p\in P_s,c_{sp}=c} z_{spi}) && \forall c \in C\\
\textstyle\sum_{p=7}^{10} x_{sp} &\leq 1 + E_s && \forall s \in S\\
\textstyle\sum_{s\in S}\sum_{p\in P_s}\sum_{b\in B} w(c_{sp},b)x_{sp} &= W^{max}\\
\textstyle\sum_{i\in I} s_i &= S^{min}\\
\textstyle\sum_{(s,p)\in U} x_{sp} &= U^{max}\\
\textstyle\sum_{s\in S}\sum_{p\in P_s,c_{sp}=c(i)} z_{spi} + s_i &\geq C^{min}_i y_i, && \forall i \in I\\
E_s &\in \mathbb{N} && \forall s \in S\\
I^{max}_c, S_c &\in \mathbb{N} && \forall c \in C\\
s_i &\in \mathbb{Z}^+ && \forall i \in I\\
\text{(x,y,z) satisfy (1)-(9)}
\end{aligned}$$

5 The Computational Results

The computational results for the academic year 2005-2006 are given in this section. This academic year was divided into six blocks. Blocks 1 & 2, blocks 3 & 4 and blocks 5 & 6 were scheduled simultaneously.

In all blocks the meetings were on Tuesday morning, Tuesday afternoon, Wednesday morning and Wednesday afternoon. Every morning and afternoon was split into two parts. So both blocks contained eight time slots. More details about the input are given in Table 3. The abbreviation wl stands for workload.

The number of students that requested workload in blocks 1 & 2 was 356 and the total workload they requested was 1416. Hence, for each block, an average of two courses of the preference list of 10 courses had to be assigned. Note that the large number of urgencies in blocks 1 & 2 can be explained by the fact that first year students are preassigned to courses, because they are not able to make a choice themselves.

Table 3. Input information for academic year 2005-2006

Blocks	$\|S\|$	$\|C\|$	$\|I\|$	$\|U\|$	offered wl	requested wl
1 & 2	356	51	79	590	1504	1416
3 & 4	328	64	88	279	1545	1288
5 & 6	302	58	89	151	1544	1333

The models introduced in Section 4 are solved by the standard IP solver CPLEX 10.0. The computations are done on an Intel Pentium M, 2.0 GHz processor with 1.0 GB internal memory. The values of the weighting factors were $W_p = 10, W_s = 1$ and $W_e = 100$. The results for the academic year 2005-2006 are given in Table 4. What can be noted is that the computation time of CPLEX is negligible.

Table 4. Results for the academic year 2005-2006

	block 1 & 2	block 3 & 4	block 5 & 6
Runtime CPLEX (s)	1.38	1.53	1.67
U^{max}	439	273	134
S^{min}	0	0	0
W^{max}	1369	1261	1300
average position	3.30	3.64	3.87
bad positions	8	16	39

In blocks 1 & 2 a requested workload of 47, in blocks 3 & 4 a requested workload of 27 and in blocks 5 & 6 a requested workload of 33 could not be assigned. Especially in blocks 1 & 2 this is caused by the small difference between the requested and offered workload. However, the main causes are preference lists for which it was impossible to assign the requested workload. Some examples of such wrongly chosen preference lists are:

- an empty preference list, because students didn't hand it in on time.
- a preference list with less than 10 courses.
- a preference list with not enough different time slots in one of the two blocks.
- a preference list with the same course on more positions. There was even a student with ten times the same course on his preference list.

If all students would hand in a preference list with 10 courses and enough different time slots, then in blocks 1 & 2 only five students would not be assigned to their requested number of courses, in blocks 3 & 4 and blocks 5 & 6 only three students.

Table 4 also shows that in blocks 1 & 2 only 439 out of 590 urgency requests could be assigned. This can be explained by the fact that in these blocks all courses on the preference list of first year students are set as urgent. Most of those preference lists contain 6 suitable urgent courses of which at most 4 are assigned. This means at least two not assigned courses with an urgency for each first year student.

The average position denotes the average of the positions of all courses assigned to a student. On average students request a workload of 4, which mostly corresponds with four courses. Hence, it can be concluded that students get a lot of courses which are on top of their preference list. A bad position is a course

assigned to a student who has the course on 7th up to 10th position on its preference list. Also from the number of bad positions it can be concluded that the courses assigned to students are on top of their preference lists.

6 Conclusions

We have formulated, analyzed and solved a real-world timetabling problem that showed up at the department of Industrial Design of the TU Eindhoven. Our successful approach was based on an Integer Linear Programming formulation. The running time that CPLEX needs for solving the resulting instances is negligible.

The administration and the students of the department of Industrial Design were highly satisfied with the timetables generated by our program. Most students now receive courses that are on top of their preference lists. There still are a few students who are not satisfied, but in most cases this turned out to be solely their own fault: they failed to specify correct preferences in the correct format.

References

1. R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: theory, algorithms and applications.* Englewood Cliffs: Prentice Hall, New Jersey, 1993.
2. V.A. Busam. *An algorithm for class scheduling with section preference.* Communications of the ACM, 10(9), 567–569, 1967.
3. E. Cheng, S. Kruk, and M. Lipman. *Flow formulations for the student scheduling problem.* In Practice and Theory of Automated Timetabling IV, Burke and De Causmaecker, Lecture Notes in Computer Science, Vol. 2740 Springer-Verlag, 2002.
4. D. de Werra. *An introduction to timetabling.* European Journal of Operations Research 19, 151–162, 1985.
5. S. Even, A. Itai, and A. Shamir. *On the complexity of timetable and multicommodity flow problems.* SIAM Journal of Computing 5, 691–703, 1976.
6. R. Feldman and M.C. Golumbic. *Constraint satisfiability algorithms for interactive student scheduling.* In Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 89), 1010–1016, 1989.
7. M.R. Garey and D.S. Johnson. *Computers and intractability - a guide to NP-completeness.* W.H. Freeman and Company, San Fransisco, 1979.
8. G. Laporte and S. Desrochers. *The problem of assigning students to course sections in a large engineering school.* Computational Operations Research 13, 387–394, 1986.
9. I. Miyaji, K. Ohno, and H. Mine. *Solution method for partitioning students into groups.* European Journal of Operations Research 33, 82–90, 1981.
10. G.C.W. Sabin and G.K. Winter. *The impact of automated timetabling on universities - a case study.* Journal of Operations Research Society 37, 689–693, 1986.
11. A. Schaerf. *A survey of automated timetabling.* Artificial Intelligence Review 13(2), 87–127, 1999.
12. G. Schmidt and T. Ströhlein. *Timetable construction - an annotated bibliography.* The Computer Journal 23, 307–316, 1980.
13. A. Tripathy. *Computerised decision aid for timetabling - a case analysis.* Discrete Applied Mathematics 35(3), 313–323, 1992.

A Max-Flow Model of Problem Formulation No. 1

Full details of the definition of this network flow problem will be given in the full version of this paper.

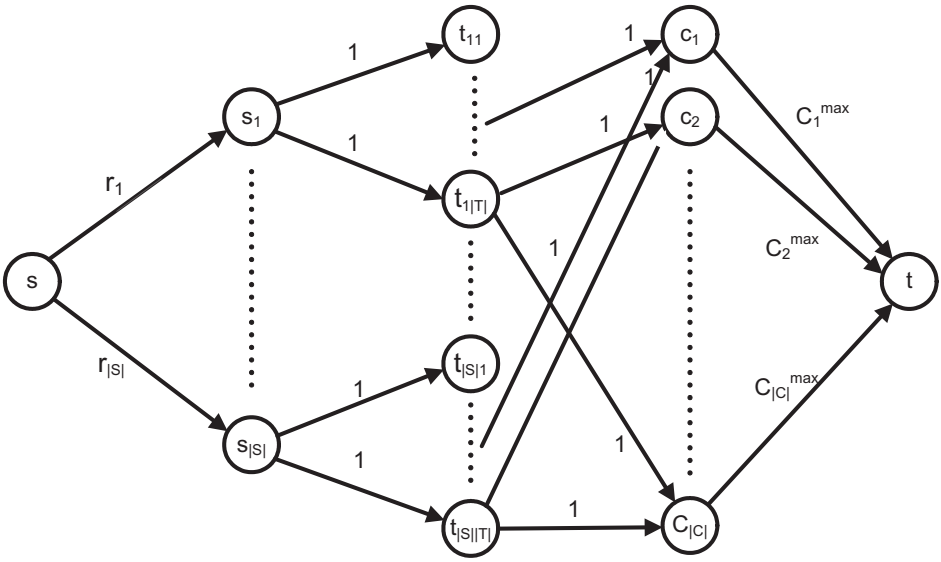

Fig. 1. The network flow model

B Some NP-hardness Results

The timetabling problem defined in Subsection 3.2 is an NP-hard problem. We prove this by identifying two independent NP-hard subproblems. Both subproblems result from adding one additional constraint to the problem formulation No. 1.

In the first subproblem, the additional constraint are lower bounds on the number of students in the courses. There are no time slots, there is only one section for each course c with a minimum and a maximum number of participating students. The workload of all courses is one, and only one block has to be scheduled. Formally, problem P_{min} is defined as follows:

> **Instance:** A set C of courses; for every course $c \in C$ a minimum capacity C_c^{min} and a maximum capacity C_c^{max} of participating students. A set S of students; for every student $s \in S$ a preference list of some courses in C, and a number r_s of requested courses.
>
> **Question:** Does there exist an assignment such that (i) every student s

gets exactly r_s courses from its preference list, and such that (ii) for every course c the number of assigned students is either zero (if the course does not take place) or falls between the bounds C_c^{min} and C_c^{max}?

Theorem 1. *Problem P_{min} is NP-hard.*

Proof. The proof is done by reduction from the *exact cover by 3-sets* problem: Given a ground set $X = \{x_1, \ldots, x_n\}$ and a set $T = \{t_1, \ldots, t_m\}$ of 3-element subsets of X, can one select $T' \subseteq T$ such that every element of X occurs in exactly one member of T'?

From an instance of the exact cover by 3-sets problem, we construct a corresponding instance of problem P_{min} with n students $x_1, \ldots, x_n$ and with m courses $t_1, \ldots, t_m$. Every student s has a demand of one course ($r_s = 1$), and every course c has minimum and maximum capacity three ($C_c^{min} = C_c^{max} = 3$).

Assume X possesses an exact cover T'. Assign student x_s to course t_c if and only if $x_s \in t_c$ and $t_c \in T'$. Since T' is an exact cover of X, every student x_s will be assigned to exactly one course t_c. The course t_c is assigned to three students if it is in T', and to zero students if it is not in T'. This shows that the constructed instance of P_{min} is a yes-instance. The converse statement can be seen in a similar way. □

In the second subproblem, we take problem formulation No. 1 and additionally allow courses with a workload of 2. We consider a situation with only one section for each course c, only a single block, and without any time slots. (And there is no minimum capacity of courses.) Problem P_{wl} is defined as follows:

Instance: A set C of courses; for every course $c \in C$ a workload $wl_c \in \{1, 2\}$ and a maximum capacity C_c^{max} of participating students. A set S of students; for every student $s \in S$ a preference list of some courses in C, and a desired workload r_s.

Question: Does there exist an assignment such that (i) every student s gets courses with a total workload r_s from P_s, and such that (ii) for every course c the number of assigned students is at most C_c^{max}?

Theorem 2. *Problem P_{wl} is NP-hard.*

Proof. The proof is done by reduction from the 3-SAT variant where every variable occurs exactly twice in negated and exactly twice in unnegated form. Consider an arbitrary instance of this 3-SAT variant.

- For every variable x_i, we introduce two corresponding students $st(x_i)$ and $st(\overline{x_i})$ which both request a workload of two.
- For every variable x_i, we also introduce a corresponding variable-course $C(x_i)$ which has a workload of two and a capacity of one. $C(x_i)$ is in the preference list of $st(x_i)$ and $st(\overline{x_i})$.
- For every clause c_j, we introduce a clause-course $C(c_j)$ with a workload of one and a capacity of two. Clause-course $C(c_j)$ is in the preference list of a student $st(x_i)$ (respectively $st(\overline{x_i})$) if and only if x_i (respectively $\overline{x_i}$) occurs as a literal in clause c_j.

Note that in any feasible assignment, student $st(x_i)$ (respectively student $st(\overline{x_i})$) will either do course $C(x_i)$ or the two courses $C(c_{j_1})$ and $C(c_{j_2})$ for which literal x_i (respectively literal $\overline{x_i}$) occurs in clauses c_{j1} and c_{j2}.

Assume that the 3-SAT instance is a yes-instance, and consider a corresponding satisfying truth-assignment. If x_i is set to TRUE, then we assign student $st(x_i)$ to the variable-course $C(x_i)$, and student $st(\overline{x}_i)$ to the two clause-courses that correspond to the clauses containing $\overline{x}_i$. If x_i is set to FALSE, we assign $st(x_i)$ to the courses that correspond to the clauses containing x_i, and student $st(\overline{x_i})$ to $C(x_i)$. Then each student receives his requested workload, and every course $C(x_i)$ gets only a single student. Since every clause has at most two FALSE literals, the corresponding clause-courses will get at most two students. So every yes-instance of the 3-SAT problem leads to a yes-instance of the timetabling problem.

Now assume that the constructed instance of problem P_{wl} is a yes-instance. Then every student $st(x_i)$ receives a workload of 2, which implies that the student must either be assigned to one course $C(x_i)$, or to two clause-courses $C(c_{j1})$ and $C(c_{j2})$. If student $st(x_i)$ is assigned to the variable-course $C(x_i)$, we set x_i to TRUE. If student x_i is assigned to some clause-courses, then we set x_i to FALSE. Since each clause-course $C(c_j)$ is assigned to at most two students, every clause contains at most two FALSE literals. Hence, every yes-instance of P_{wl} corresponds to a yes-instance of 3-SAT. □