Aalto University
School of Science
Degree Program of Technical Physics and Mathematics

Vendla Sandström

Scheduling of teaching resources and classes using mixed integer linear programming

- A case study in a university of applied sciences

Mat-2.4108 Independent Research Project in Applied Mathematics
Helsinki, December 10, 2012

Supervisor: Professor Raimo P. Hämäläinen
Instructor: M.Sc. Antti Toppila

Contents

1 Introduction

Activities that may overlap are scheduled in order to ensure the availability of resources shared by the activities. Typically a schedule can assign activities to take place in many different time slots. However, some schedules are better than others in that they enable more activities, reduce the peak resource capacity or satisfy customers' needs better. The quality of the schedule does also affect the everyday work of the staff, giving a great importance to what is usually considered a simple allocating task. Schedules are still done by hand in many organizations, although todays information technology would allow computer aided optimization. When the combinatorial complexity rises as organizations grow larger, bottleneck resources are overcapacited to guarantee the finding of workable schedules. By using better optimization tools, the need of excess resources could be decreased. This again allows the organization to decrease their expenses in the long run.

In this research project, mixed integer linear programming is used to create better schedules in a university of applied sciences. Education scheduling, or timetabling as it is usually called, holds a lot of complexity due to a big amount of activities and resources in use. The timetabling problem can be defined as allocating the courses to time slots and suitable rooms according to group sizes and equipment needed. The choosing of teachers for the courses can be done in advance, or it can be a part of the problem. The more wishes of teachers and students that are taken into consideration, the better the schedule is usually perceived.

The research problem is to describe the case university's timetabling problem mathematically and then to solve it using mixed integer linear programming. The aim of the research is to find a simple but effective way of solving timetabling problems in the case university. The main focus is on finding effective and satisfactory solutions, which means the solutions are feasible, can be realized in the university and gets a sufficient acceptability by the users. In timetabling, the main goal is usually feasibility and acceptability, not optimality (McCollum 2006; Daskalaki et al. 2004), because usually there are so many constraints, that when a feasible solution is found, it is considered good as well. Finding a good solution in a short time is also considered more valuable than finding an optimal solution after a long time of optimization (McCollum et al., 2010) (the schedule may already be old by that time).

The literature holds many descriptions of timetabling problems and suggestions of how to solve them (see Werra 1985 or Montana et al. 1998). Most of these are focused on solving problems in theory and not in practice (Murray et al. 2006; McCollum 2006). When creating and testing optimization algorithms, test data is usually (over)simplified, making the solving easy, while an instance of the same size in real life may be anything but easy to solve (Murray et al. 2006; De Causmaecker & Van den Berghe 2012). In this work, the usability of the results is a major issue, making simplifications undesirable.

The idea and general principles of scheduling is straightforward, but in most cases, the amount of data is large enough to create a great amount of combinatorial complexity. For small instances, the problem can be solved by direct approaches, but for most real life problems, a solving of the whole problem at once using direct approaches is impossible. For this, there is generally two types of solutions, either to divide the problem into smaller parts that can be solved one at a time or to use different heuristics to solve the problem directly (Murray et al. 2006, Zhai et al. 2010). In this research, the former approach is chosen.

The model developed in Section 2 is inspired by the methods used in the case university. The model considers the whole process of scheduling, from choosing periods to the courses to creating individual weekly schedules. There are many examples of how to allocate courses to week hours (see for instance Murray et al. 2006; Daskalaki et al. 2004; Martin 2004), but generally the models do not consider the choosing of order for the courses. That part is here added to consider the leveling of resource needs better in the long run.

The data gathering and model development has mainly been done in parallel, and feedback on the first solutions has been used to develop better models and revise the data used. The three step model crated is able to produce feasible schedules of acceptable quality. Although the case problem is to some extent simplified, the changes done are small and can be easily added to the problem. The model creates individual schedules for each week, taking into consideration the changing course patterns, including different types of lectures (with different room needs), their differing lengths and the teachers availability (part time teachers are not available all days).

The rest of this paper is structured as follows: Section 2 holds the description of a general model which then is tested in the case study presented in Section 3. The results of the case study and the usability of the model is further discussed in Section 4. Section 5 concludes.

2 Scheduling with mixed integer linear programming

This section describes a general model for optimization of teaching resources and educational activities. The optimization is in this work divided into stages, for which there are two main reasons: First, the problem is too big to be optimized at once, there is simply just too many choices to be made and the computer runs out of memory. Second, it is easier to understand and alter the outcome of a stepwise optimization; the different weekly timetables give a vague picture of how the courses are grouped into different years. It is pointless to optimize the weekly schedules before the courses are reasonably divided into periods.

Breaking a problem into parts does not necessary cause as good a solution as solving the problem at once would (Murray et al., 2006), but when it comes to timetabling, finding a good feasible solution will be enough; optimal solutions are mostly not required. The more independent the parts are, the less detrimental is the division.

The number of optimization stages used can vary; some universities use the same schedule from week to week during a period, while others have individual weekly schedules. Some use the same sequence for the courses from year to year, others may let it vary. If all parts of the optimization process are used, the path will be as follows: first the courses are divided into periods, representing all the years of study for the student groups. Next, the course lectures are assigned to different weeks, and at last, the weekly timetables are made.

In addition to the time division explained above, division according to departments or other logical entities can also be done. This is a usual approach used in university timetabling (Daskalaki et al. 2004; Martin 2004; Murray et al. 2006). Departments have traditionally a high degree of independency and little in common with each other, making a division of the overall problem quite natural. Murray et al. (2006) also point out that allowing the departments create their own schedules is important for the transparency of the process and the commitment to the final timetables.

The approach used here is not so different from the one used at the case university, the important difference is the automating of the process. Their schedules are still handmade, and the process usually takes several weeks depending on the size of the task and how well information flows in the organization. This means that automating the processes would be of great help, and a rise in the quality of the timetables would be welcome although it is not necessary.

2.1 General principle for scheduling

The general principles presented here are the same for all stages of the optimization. Individual stage differences are presented later on in the respective chapters. Here period refers to any time period, may it be terms, weeks or hours. Activities are courses or parts of courses such as lessons, but it could also include one-time events like seminars or theme days. Resources are class rooms, teachers and anything else a course may need such as cars or video projectors. Student group denotes single students or groups of students participating in the same courses. The students (customers in this case) choses courses (activities), which need resources to be performed. The schedule must allow all students to attend their chosen courses. The availability of needed resources must be considered.

The problem of scheduling includes the following types of constraints:

a) Students should have as even a workload as possible.

b) The usage of resources must be within a certain interval. (Maximum usage cannot exceed total available resources, but it may be required to be smaller as well. Resources like teachers may have a minimum usage.)

c) Every activity must be held at some point.

d) Activities have differing lengths and needed resources.

e) Every student and every resource may only have one activity booked at a time $\rightarrow$ maximum booked activities per student or resource must be in proportion to the time period. (Momentary usage can be up to 100 %, the long term average ought to be smaller.)

f) An activity may have several resources in use at once.

g) Some activities must be in a particular order, and they may have a pre-assigned period.

h) All resources are not available all the time.

Constraints are usually divided into soft and hard constraints (Smith-Miles & Lopes 2012; Müller 2009). Hard constraints are the real constraints in the optimization and soft constraints the ones making up the objective function. The hard constraints must be obeyed, breaking of the soft ones avoided if possible. Of the above listed constraints, the first one, a), is used as a soft constraint, the rest as hard constraints.

Four parameters are used for the basic structure of the model. These are:

- The time periods available, p, which may be terms, weeks or hours depending on the model used. The set of all time periods is denoted by $P = \{1, 2, 3, ..., n_p\}$, where n_p is the total number of time slots used.
- The activities to be scheduled, a, could be courses or single lectures or other events. The set of all activities is denoted by $A = \{1, 2, 3, ..., n_a\}$, where n_a is the total number of activities to be scheduled.
- The groups of students, s. All students in a student group have all courses in common, and a student group is principally made up of different combinations of a degree program's major and minor combinations. Combinations of elective courses can also be considered. The set of student groups in denoted by $S = \{1, 2, 3, ..., n_s\}$, where n_s is the total number of groups.
- The resources types needed, r_i. These are rooms, teachers or any other resources needed. The set of a resource type i is denoted by $R_i = \{1, 2, 3, ..., n_{ri}\}$, where n_{ri} is the total number of resources of type i.

Resources have a maximum usage limit $R_{i,MAX}(r_i)$, for example, the maximum teaching hours a teacher can have a week. The maximum limit could be the same for all resources or there could be individual limits for each resource. Some resources may also have a minimum usage limit $R_{i,MIN}(r_i)$. For instance, teachers may have a required minimum amount of teaching each week.

In addition to the parameters described above, there are some parameter matrices describing student subscriptions and resources needed for the activities.

Student group study modules (the activities a group of students attend):

$$StudyModule(a, s) = \begin{cases} 1 & \text{if the activity } a \text{ is included in the studies of group } s \\ 0 & \text{otherwise} \end{cases}$$

The total hours of a resource r_i required by an activity in a period:

$$Resource_i(a, r_i) = \begin{cases} n & \text{if the activity } a \text{ needs resource } r_i \text{ for } n \text{ hours in a period} \\ 0 & \text{otherwise (the resource will not be needed)} \end{cases}$$

For resources that are partly available:

$$NoResource_i(p, r_i) = \begin{cases} 1 & \text{if the resource } r_i \text{ is unavailable at time } p \\ 0 & \text{otherwise.} \end{cases}$$

The activities may be required to be in a particular order. The set of activities to have an order is $O = \{1, 2, .., n_o\}$, where n_o is the total number of ordered pairs and o denotes the individual elements of O.

$$OrderBefore(o) = a_1$$
$$OrderAfter(o) = a_2$$

where a_1 and a_2 are elements in A.

Activities may also be required to be in a certain period:

$$\text{s.t. } ChosenPeriod(a) = n : \quad \chi(a, n) = 1$$

The decision variable, here noted as $\chi(a, p)$, may be Boolean (step 1 and 3) or integer (step 2) depending on the situation. The variable chooses periods for the courses. The specific decision variables used at each stage are presented in the respective sections, as well as additional, situation specific constraints and the objective functions. The hard constraints of the general model are listed below.

All activities must be held in some period (constraint c)):

$$\forall\, a \in A, \quad \sum_{p \in P} \chi(a, p) = 1 \tag{1}$$

The usage of resources must be below the allowed maximum level (constraint b)):

$$\forall\, p \in P, \quad \forall r_i \in R_i, \quad \sum_{a \in A} \chi(a, p) Resource_i(a, r_i) \le R_{i,MAX}(r_i) \tag{2}$$

In case there is a minimum amount the resource must be used, the following equation will be needed (constraint b)):

$$\forall\, p \in P, \quad \forall r_i \in R_i, \quad \sum_{a \in A} \chi(a, p) Resource_i(a, r_i) \ge R_{i,MIN}(r_i) \tag{3}$$

If a resource is not available at some point ($NoResource_i(p, r_i) = 1$), then there can be no activity requiring that resource at that point (constraint h)):

$$\forall\, p, r_i \text{ such that } NoResource_i(p, r_i) = 1 : \quad \sum_{a \in A} \chi(a, p) Resource_i(a, r_i) = 0 \tag{4}$$

For activities having a particular order, the following set of equations can be used (constraint g)):

$$\forall\, o \in O, \quad \sum_{p_1 \in P} p_1 \chi(OrderBefore(o), p_1) + 1 \le \sum_{p_2 \in P} p_2 \chi(OrderAfter(o), p_2) \qquad (5)$$

The constraint tells that if there is a required order, the period p_1 of the first activity (left side of equation) must have a smaller period number than that of the later activity (right side of equation).

The pre-assigned periods gives the last equations (constraint g)):

$$\forall\, a \text{ such that } ChosenPeriod(a) > 0 : \quad \chi(a, ChosenPeriod(a)) = 1 \qquad (6)$$

Here, no separate equations have been set for constraints d), e) and f). These are either a part of the equation sets (2) and (3) or they will generate additional equations in the stages presented below.

2.2 Choosing terms for the courses

Here the characteristics of the first stage of scheduling are described. The activities at this level is the courses, the time periods terms. The constraints needed are equations (1)–(6), but the time influence is a bit different. When the whole study path of a student group is optimized at one time, the courses will be divided into several years. Then, all courses in the first term of a year must be considered when calculating the maximal usage of a resource. For instance, if there are two terms in a year and the study program will last four years, then there is a total of 8 periods where the courses can be. But when considering the use of resources, the courses in period 1, 3, 5 and 7 will be using the same resources. Thus, the four terms must be put into the same constraints. This affect equation sets (2) and (3). No new equation types are needed.

The decision variables at this stage are denoted $x(a, p)$, where

$$x(a,p) = \begin{cases} 1 & \text{if the activity } a \text{ is in period } p \\ 0 & \text{otherwise} \end{cases}$$

The main goal of the optimization is to obtain as even a workload for the students as possible. Since the total amount of work is constant, additional variables are needed. The auxiliary variables are named y_i and punishes for deviations from the goal amount of credits, denoted $goal_i$. Since courses have different credits (and thus different workloads), the credit amount $cr(a)$ must be added to the equation. The amount of auxiliary variables needed depends on the desired objective function (see below). All y_i:s are determined by the following equations:

$$\begin{aligned} &\forall\, p \in P, \quad \forall\, s \in S, \\ &\quad y_i(p,s) \geq \textstyle\sum_{a \in A} StudyModule(a,s)x(a,p)cr(a) - goal_i \\ &\quad y_i(s,p) \geq 0 \end{aligned} \tag{7}$$

A weighted objective function is created according to preferences. The weights ω_i are used to emphasize parts of the equation more or less than others by increasing or decreasing the weights. The weights can be determined by logic or by testing or any combination of both. The same concerns the number of auxiliary decision variables y_i needed. In the case study later in this work, the testing approach has primarily been used. The general objective function frame is:

$$min \sum_{p \in P} \sum_{s \in S} \sum_{i} \omega_i y_i(p,s) \tag{8}$$

2.3 Weekly lecture allocations

At this stage, lectures are allocated to weeks. This stage is only needed when the weekly schedules differs from each other. Now, the basic activity unit is no longer courses but lectures. A lecture is here defined as one hour of teaching, and several lectures next to each other (and of the same course) is called a class. Class lengths can vary, and this must be considered in the optimization. The output of stage 1, the courses in each term, is used as input data. In practice this means that the set of courses is here less or equal to the amount used in the first stage. In the case of two terms each year we get $A_{autumn} \cup A_{spring} = A$ and $A_{autumn} \cap A_{spring} = \emptyset$, that is, a course is either in the fall or the spring term, but not in both.

The decision variable, $z(a,p)$, is the amount of classes a week.

$$z(a,p) = \; n, \text{if the class } a \text{ is held } n \text{ times at week } p$$

The constraints needed from the general model are (1)–(3), the need of constraint (4) depends on the situation. Class lengths must be considered in all constraints. For instance, the total amount of classes during the weeks must equal the amount of course lectures per class length. Then, the constraint saying every activity must be held (equation (1)) is now:

$$\forall\, a \in A, \quad \sum_{p \in P} z(a,p) = TotalHours(a)/ClassLength(a) \tag{9}$$

where $TotalHours(a)$ denotes the total lecture hours of a course and $ClassLength(a)$ the amount of consecutive lectures in a class.

Since we do not want all lectures of a course during one week, but spread out evenly during the term, the amount of course classes a week must be within a certain interval.

$$\begin{aligned} &\forall\, p \in P, \quad \forall\, a \in A, \quad z(a,p) \geq minHours(a,p) \\ &\forall\, p \in P, \quad \forall\, a \in A, \quad z(a,p) \leq maxHours(a,p) \end{aligned} \tag{10}$$

where $minHours(a,p)$ denotes the minimum amount of classes of a during week p, and $maxHours(a,p)$ the maximum amount. This equation set replaces the eventual need of equations (5) and (6) at this stage.

The workload from week to week should be as even as possible for the students. The goal amount of week hours for each student group is denoted by $goal_i$. Again, auxiliary variables, y_i punishing for deviations from the average, are needed to formulate the objective function.

$$\begin{aligned} &\forall\, p \in P, \quad \forall\, s \in S, \\ &\quad y_i(s,p) \geq \textstyle\sum_{a \in A} StudyModule(a,s) z(a,p) ClassLength(a) - goal_i \\ &\quad y_i(s,p) \geq 0 \end{aligned} \tag{11}$$

Now, the weighted objective function (weights denoted by ω_i and acquired in the same way as described in section 2.2) is:

$$min \sum_{p \in P} \sum_{s \in S} \sum_{i} \omega_i y_i(s,p). \tag{12}$$

2.4 Creating the schedule

This last stage is the most important since the final timetable is done here. If optimization has been done according to the previous stages, that output is then used as input data here (that is, which courses and/or classes are to be allocated during a week). Equations (1)–(3) are always needed at this stage and (4)–(6) can be used according to the situation.

At this stage two decision variables are needed. The main decision variable, $w(a,p)$, denotes the first hour of a class. An auxiliary decision variable, $y(a,p)$, denotes the following class hours (if a class is longer than one hour).

$$w(a,p) = \begin{cases} 1 & \text{if the class } a \text{ starts at time } p \\ 0 & \text{otherwise.} \end{cases}$$

$$y(a,p) = \begin{cases} 1 & \text{if the class } a \text{ is held at time } p \\ 0 & \text{otherwise} \end{cases}$$

If all classes were of equal length, say two hours and the day is divided into a couple of two-hour blocks, only one decision variable would be enough. Then the equations (13)–(15) below would not be needed. Here the more difficult case of differing class lengths is presented.

For one hour long classes, no extra constraints are needed. If the class length is more than one hour, classes must not be divided into two days. For instance, if a class is two hours long, it cannot begin at the last hour of a day. The set $T(a) \in P$ denotes the hours when a class cannot begin.

$$\forall\, a \in A, \quad \forall\, p \in T(a), \quad w(a,p) = 0 \tag{13}$$

The class hours must be next to each other. The number of consecutive hours is denoted by n_i (first hour not included). For instance, a three hour class has the n_i value 2. The set of consecutive hours is denoted by $I(a)$ for which holds $i \in I(a) = \{1, ..., n_i(a)\}$. Now, the following set of equations state that if period p is chosen as the first hour of an activity a, the following $n_i(a)$ hours must also be reserved for that activity. The objective function ensures that no extra y values are turned to 1 than absolutely necessary.

$$\forall\, a \in A, \quad \forall\, p \in P, \quad \forall\, i \in I(a), \quad w(a,p) \leq y(a,p+i) \tag{14}$$

Since there are two decision variables, for each activity, only one decision variable is allowed to differ from zero at each given point in time (otherwise two activity hours would be held atop of each other).

$$\forall\, a \in A, \quad \forall\, p \in P, \quad w(a,p) + y(a,p) \leq 1 \tag{15}$$

In addition, a student group can only have one lecture at a time (constraint e)):

$$\forall\, s \in S, \quad \forall\, p \in P, \quad \sum_a (w(a,p) + y(a,p)) StudyModule(a,s) \leq 1 \tag{16}$$

The objective is to minimize the amount of free periods during the middle of the day for the student groups. For this, some new index sets are needed. The set $D = \{1, .., 5\}$ denotes the days in a week and $H = \{1, ..., n_h\}$ denotes the timeslots during a day, n_h being the sum. By using weights $\mu(h)$ $(h \in H)$ shaped as a piecewise linear approximation of an upward opening parabola, hours at the beginning or the end of the day are punished for more than hours in the middle of the day. Thus, as much as possible of the hours in the middle of the day will be used, creating a compactness of the schedule. The objective function can be written as:

$$min \sum_{a \in A} \sum_{d \in D} \sum_{h \in H} \mu(h)(w(a, h + (d-1)n_h) + y(a, h + (d-1)n_h)) \tag{17}$$

3 Optimizing the timetable in a university of applied sciences

The model developed in Section 2 was used for optimizing the timetables in Hämeen Ammattikorkeakoulu (HAMK), a university of applied sciences. Timetables were done for house B, where four different degree programs arrange their activities. The chosen case is large enough to include all aspects of the model developed in Section 2, and the House B holds an average amount of activities compared to other buildings in HAMK. If the model produces good solutions to this problem, it will most probably find good enough solutions to other timetabling instances in the university as well.

According to the schedulers, producing good timetables is hard, since there are too few large lecture rooms and large ADP lecture rooms in House B. At the moment, schedules are first made according to student groups and teachers. The students ought not to have any free periods during the day, so the schedule must be compact. The teachers may have seminars, meetings, lectures held in the evening or they may be working only part-time, all of which limits the time they are available. In addition, they may have preferences on how their lecture hours are grouped into teaching events. All of these preferences cannot usually be taken into consideration, since they may lead to infeasible schedules. Restrictions (unavailability) are always considered, preferences or whishes are considered when possible. Rooms needed for the lectures are taken into consideration when the best possible schedule for teachers and students is done. Schedules are usually not changed anymore at this state, so if the need for a certain type of room is greater than the amount available, a shortage of that room type is perceived. Then, a room is searched for in another building.

When looking at the amount of different room types needed on average and comparing to the existing rooms, there seems to be enough rooms of all kinds to satisfy the demand. But if the demand is too uneven, shortages will occur. This may be the result when needed room types are not considered in the first steps of scheduling.

The aim in this case is to test whether computer aided optimization can generate better schedules than the human schedulers. If good enough schedules can be made, is the amount of work needed small enough to make it an agreeable method for scheduling in the future? By adding constraints, the optimization can be more and more customized, but when the customization of the optimization tool consumes more time than the original way of scheduling, the optimization will not necessary be worth the effort.

Data for the scheduling problem was gathered primarily from the university web sites, but for the parts when information was not detailed enough or missing, additional information was asked of the schedulers in house B. The case holds three types of simplifications. First, it is assumed that the activities in the other buildings do not affect the activities in house B, and second, in the first two stages, all resources are presumed available all time. The third simplification states that there are no alternative resources to use (for instance a larger lecture room or another teacher). If taken into consideration, the first two simplifications would decrease the amount of solutions, the last one increase them.

The data is gathered into an Excel-workbook and formatted for the optimization. The optimization itself is done using the integer programming solver CPLEX. The solving program imports the needed information from the workbook, and exports the results back to the same workbook. Since the results are binary (or integer) matrices, the information is then processed into a more easily understandable form (see Figure 5 on page 21).

3.1 Results

In this section the results of the optimization are presented. The model in Section 2 is of a general nature, so first, the parameters and restrictions used in this optimization are explained. The model is tested stepwise using the same order as in Section 2. The results are computed with CPLEX on a computer with an Intel Core Quad 3.2 GHz and 4 GB RAM. The code used can be seen in appendix B.

At the first stage, terms are chosen for the courses. The goal of the optimization is to have a workload for the students as close to 30 credits each term as possible. The resources considered are teachers and room types (for instance, lecture rooms for 25 students or ADP rooms for 35 students). There is a maximum usage of both resources but no minimum. All parameters and decision variables are summarized in Table 7, appendix A.

There are two unknowns to decide by trial and error. The first is the maximum percentage the resources can be used, and it ought to be somewhere between 70 % and 100 % (Pennanen, 2004). The value must give out data that makes it possible to find solutions at the later stages as well. The values 80 % for rooms and 83 % for teachers gave feasible solutions (including the later stages as well). The second unknown is the exact formulation of the objective function - the amount of auxiliary variables and their weights must be chosen. Only using one variable (y_1) was not enough, since a deviation of 5 credits is considered far worse than five times a deviation of 1 credit. A second variable y_2 was added, punishing double up for any workload bigger that 31 credits. This produced better results, but now negative deviations of up to -6 were found. Negative deviations are not as bad as positive ones, but big negative deviations are not recommended either. A third variable y_3 was added, punishing for deviations of less than 28 credits. This produced acceptable results. The final auxiliary variables and the objective function are presented below in Table 1 and graphically in Figure 1.

Table 1: Parameter values for the objective function

Parameter	$goal_1$	$goal_2$	$goal_3$	w_1	w_2	w_3
Value	30	31	28	1	2	0.5

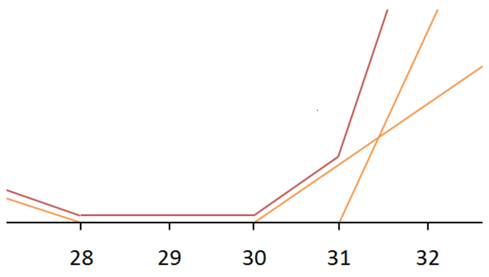


Figure 1: The objective function of the first stage. The orange lines denotes the different parts of the objective function, the red line is the sum of the orange ones.

Different results from the first stage are in Table 2. The gap is the difference between the best node, that is the best objective function value achievable and the best integer (solution) found. If the gap reaches zero, an optimal solution has been found. In the first stage, an optimal solution was found in less than two minutes. A first solution was found in less than a second, although that solution was far from optimal. In less than a minute's time, the optimal value of 43 is reached, but another half a minute is needed to prove the value optimal.

Table 2: Results of the first stage

Time [s]	**Value**	**Gap** [%]	**Feasible**
57	43	3.3%	yes
96	43	0%	yes

Analyzing the optimized results, they could be considered quite good, which means the student workloads are mostly even. The greatest deviation from the goal of 30 points each term is ±3, but for most parts the deviation is only one point in either direction. Slightly less than half (48 %) of the term points deviates from the goal. For some student groups, the workload does not deviate from the goal amount at all, while for other groups it deviates in every term. Figure 5a on page 21 shows a screenshot from the data worksheet showing the results of the optimization for one of the student groups.

The optimized workload is fluctuating a bit more than the one the schedulers make, but then again, in practice the courses are divided into smaller parts and they may be held in up to five different terms at the most. In this optimization, some of the largest courses are divided into two parts, but otherwise one course is held during one term and not stretched out into a year or two. The habit of stretching out courses stems partly from problems finding feasible schedules, partly from old practice.

Figure 2 shows the average need of the different room types each term. The need is given as a percentage of the total amount of rooms and time available. Note that the amount of rooms of a room type varies between one and five rooms. For most room types the division of lecture hours into autumn and spring terms is quite even, but for some it varies more and is at the most growing the double. In the optimization, a usage constraint of max 80 % is used to guarantee that solutions can still be found in the following steps. For two of the room types the average need is more than 80 %. In both cases, the need for larger classes of the same kind is low enough to shift capacity to the smaller category.

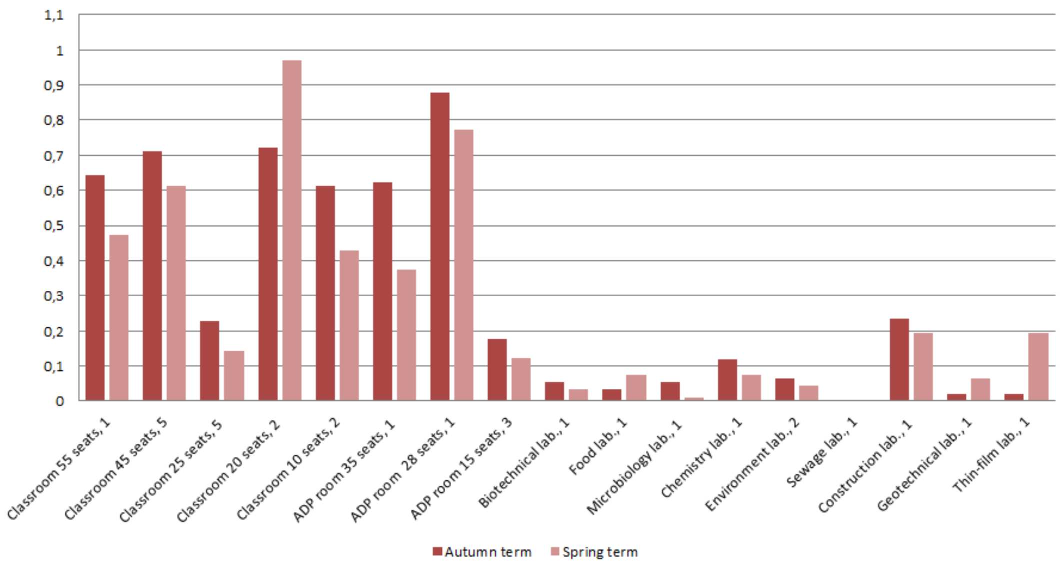


Figure 2: Average need of different room types in percent. The classification on the x-axis explains the room type, number of seats, and last, how many rooms of that type there are in House B.

The result of the optimization is quite robust, changing the constraints of rooms and teachers a little (less than 10 %) do not affect the outcome. The teachers' workload could still be raised and all rooms have not been used in the optimization. The rooms not used in the optimization are rooms that statistically are unnecessary, and adding them to the constraints do not actually improve the results at all at this stage. Changing the order constraints on the other hand affects the outcome more easily, depending on if the changed constraint has been an active constraint or not. In the iterative testing stage, removing an unnecessary tight order constraint made the goal value drop by more than 200, to a third of its former value. This means that questioning the constraints of order seems to be the easiest way of creating better schedules, instead of building more classrooms or hiring more teachers.

The input data for the second step of optimization is partly the output data of the first one. The only new information needed for the next stage is the total amount of lecture hours for each course, the class lengths and the minimum and maximum hours for each week. The rest of the needed data can be generated from the one used in the first stage and from the outcomes. The used parameters and variables are summarized in Table 8, appendix A.

Again, there are two unknowns that need to be determined, the maximum resource usage and the objective function. A maximum usage of 90 % of total available room and less than 30 lecture hours per week for the teachers produces feasible solutions. For the objective function, only one dummy decision variable (punishing for deviations above the average) is not enough, since a big deviation is considered worse than several small ones. By adding y_2 punishing for deviations more than 5 hours over the average week time, better solutions are found. The average week hours for a student group is denoted by $awH(s)$. The dummies and objective function weights used are shown below in Table 3 and illustrated in Figure 3.

Table 3: Parameter values for the objective function

Parameter	$goal_1$	$goal_2$	w_1	w_2
Value	$awH(s)$	$awH(s)+5$	1	2

The results of the optimization is shown in Table 4. This time a longer optimization time did not improve the goal value, although the gap became smaller. The average number of lecture hours a week in the outcome is quite stable, the standard deviation is 3.7 % of the average. The work placements in the beginning or end of the term increase the average lecture hours, making it impossible to have a completely even allocation of the lectures. The outcome can be considered acceptable.

In the third stage, the weekly schedules are made. Again, the outcomes of the previous stages are used as input data for the last stage. In addition to earlier data, the time some teachers are unavailable is considered as well. Since the calendar of every week is done individually, only two different weeks are presented here. The weeks chosen have

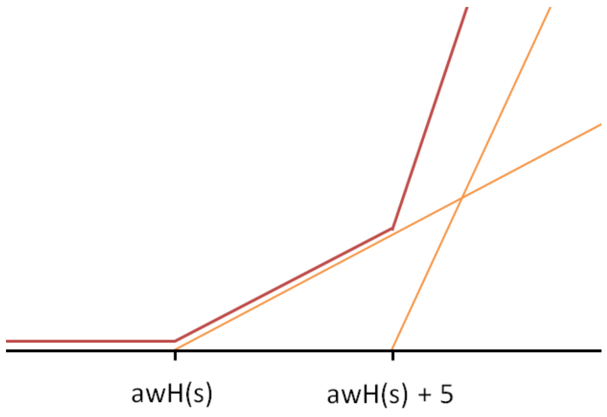


Figure 3: The objective function of the second stage. The orange lines denotes the different parts of the objective function, the red line is the sum of the orange ones.

Table 4: Results of the second stage

Time [s]	**Value**	**Gap [%]**	**Feasible**
200	2334	0.6%	yes
2000	2334	0.5%	yes
3198	2334	0.4%	process out of memory

both lecture amounts above the average; week 8 is actually the most "busy" one. If results for the more "crowded" weeks can be found, the finding of feasible schedules for the weeks with fewer lectures will most probably not be a problem. The parameters and variables are shown in Table 9, appendix A.

Now the maximum usage of resources can be up to 100 %. There are nine timeslots each day, and the weighting function $\mu(h)$ is chosen to be an upward opening parabola, illustrated in 4. The values used are listed in Table 5. This function produced so good results that no further trial and error was needed.

Table 5: Parameter values for the objective function in the third stage

Parameter	$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$	$\mu(9)$
Value	4	2	1	0	1	2	4	8	12

Table 6 holds the results of optimization in the third stage. For both weeks, a first solution is found quite fast, but to prove a solution optimal takes several hours if not the process gets out of memory (week 8). Then again, solutions for the "easier" weeks are found faster.

Analyzing the calendars created, both calendars can be considered good. That means

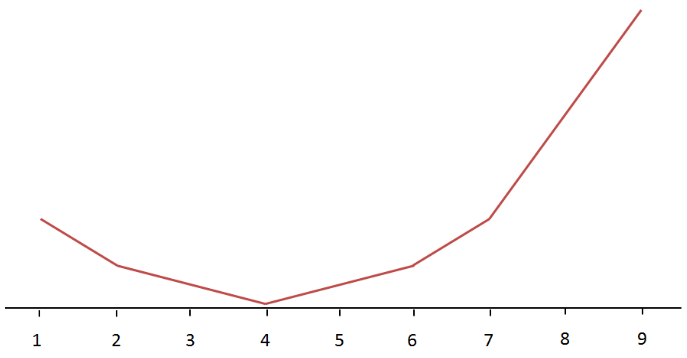


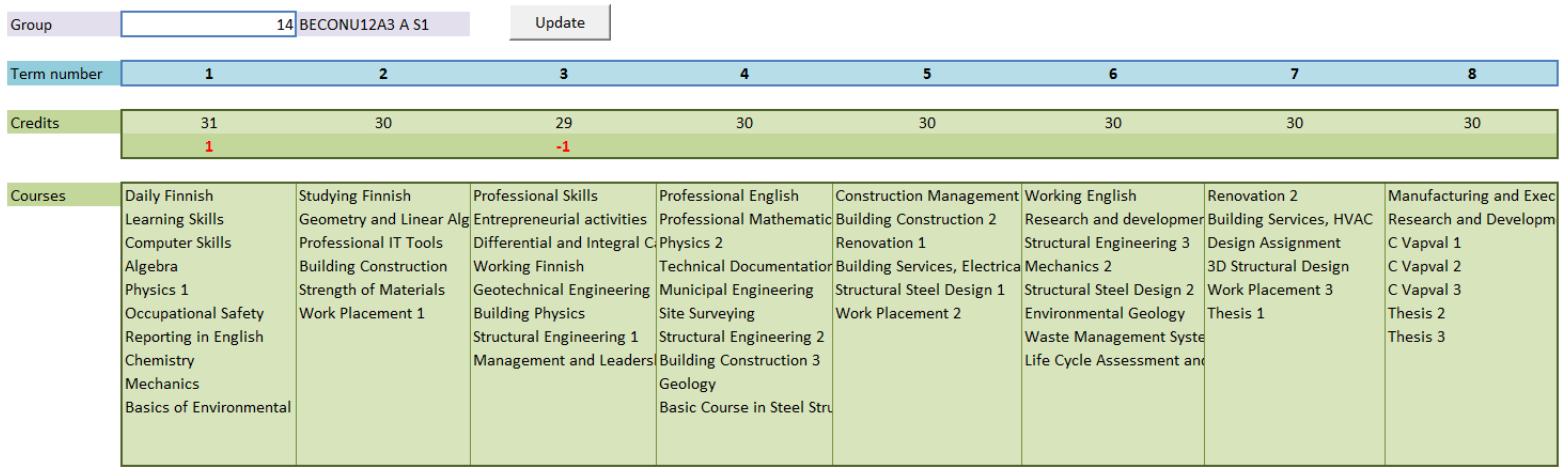
Figure 4: The objective function of the third stage.

Table 6: Results of the third stage

Week	**Time** [s]	**Value**	**Gap** [%]	**Feasible**
Week 8	145	981	0.92%	yes
Week 8	6943	981	0.54%	yes
Week 11	30	937	0.21%	yes
Week 11	6909	937	0.00%	yes

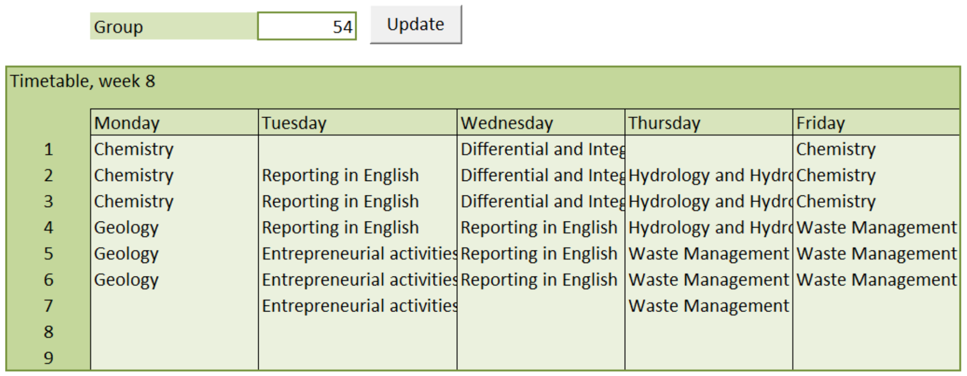
only sporadic free periods occur, there are about one a week per every tenth group. The days are of quite equal length, giving the students a mostly equal workload during the week. Figure 5b on page 21 shows a screenshot of the timetable view.

Taking into account the results of all three steps, the algorithm can be said to function quite well considering its simplicity. The results are not optimal but acceptable, which means that the main goal of finding feasible solutions has been achieved. From the students' viewpoint, the timetable is as good as it has been before, but since the automated optimization did not have problems with sufficiency of rooms, the outcome can be considered considerably better than the handmade schedules.



Group	14	BECONU12A3 A S1	Update					
Term number	1	2	3	4	5	6	7	8
Credits	31	30	29	30	30	30	30	30
	1		-1					
Courses	Daily Finnish	Studying Finnish	Professional Skills	Professional English	Construction Management	Working English	Renovation 2	Manufacturing and Exec
	Learning Skills	Geometry and Linear Alg	Entrepreneurial activities	Professional Mathematic	Building Construction 2	Research and developmer	Building Services, HVAC	Research and Developm
	Computer Skills	Professional IT Tools	Differential and Integral C	Physics 2	Renovation 1	Structural Engineering 3	Design Assignment	C Vapval 1
	Algebra	Building Construction	Working Finnish	Technical Documentatior	Building Services, Electrica	Mechanics 2	3D Structural Design	C Vapval 2
	Physics 1	Strength of Materials	Geotechnical Engineering	Municipal Engineering	Structural Steel Design 1	Structural Steel Design 2	Work Placement 3	C Vapval 3
	Occupational Safety	Work Placement 1	Building Physics	Site Surveying	Work Placement 2	Environmental Geology	Thesis 1	Thesis 2
	Reporting in English		Structural Engineering 1	Structural Engineering 2		Waste Management Syste		Thesis 3
	Chemistry		Management and Leaders	Building Construction 3		Life Cycle Assessment anc		
	Mechanics			Geology				
	Basics of Environmental			Basic Course in Steel Stru				

(a) Terms



Group	54	Update			
Timetable, week 8					
	Monday	Tuesday	Wednesday	Thursday	Friday
1	Chemistry		Differential and Integ		Chemistry
2	Chemistry	Reporting in English	Differential and Integ	Hydrology and Hydrc	Chemistry
3	Chemistry	Reporting in English	Differential and Integ	Hydrology and Hydrc	Chemistry
4	Geology	Reporting in English	Reporting in English	Hydrology and Hydrc	Waste Management
5	Geology	Entrepreneurial activities	Reporting in English	Waste Management	Waste Management
6	Geology	Entrepreneurial activities	Reporting in English	Waste Management	Waste Management
7		Entrepreneurial activities		Waste Management	
8					
9					

(b) Timetable

Figure 5: Screenshots of the representation of the results. Figure (a) shows the terms chosen for the courses of group 14, construction engineers, and (b) shows the timetable made for the second year construction engineering students for week 8.

4 Discussion

The results of the optimization were good (feasible but not perfect), and if the optimization would be done together with the schedulers, adding the last missing information, usable results could be generated. The model seems to be sufficient for the case organization and with some additions to the objective function, even better results could be produced. A cogent reason not to implement the model at this stage is the user interface; it requires that the user is accustomed to modeling and all the parts of the model. The CPLEX-Excel combination is too clumsy and heavy to be used in practice. If a better user interface would be created, the model could be used as it is.

A so called pre-enrollment approach was used in this work. This means schedules are done according to curricula, not actual student enrollments. McCollum (2006) points out that there are several reasons for making timetables this way; there is no guarantee that feasible timetables exist after enrollment, and students usually need to know timetables before choosing which (parts of) courses to participate in. But, pre-enrollment schedules do not mean student requests are not considered at all, personal curriculum information can be gathered, and if they are up to date, timetables can be optimized to fit as many students as possible. In this work, feasible schedules for all groups have been weighted equally, but weighting could also be done according to group size. The more popular major subjects had five times more students than the less popular ones, and the weighting of the objective function could be done accordingly.

McCollum 2006 states, that one of the main reasons feasible timetables are still found despite the great flexibility for students and teachers to choose timetables to their liking, is that currently, universities uses about 30 percent of their space effectively. That is, during a normal workday (for instance 8 or 10 hours long) the hours a room is used in average. The case used here is no exception. In house B, the average use of classrooms is 33 percent, but it can be noted that the usage of different laboratories is only 7 percent, while the use of general classrooms and ADP classes is 46 percent. Solutions were found to instances using 16 rooms instead of 20, raising the effective use of space to 58 percent (laboratories not included). An interesting observation is that the results of the optimization with less space available were about the same or the same as the results when all space was available.

It is clear that all excess rooms cannot be removed: If the utilization of rooms were 100 percent, feasible schedules would most probably not be found. But the usage can still be close to 100 percent for single rooms without making the finding of schedules impossible, although it will be much harder (see for instance Müller (2009)). Additional research could be done investigating how the amount of available rooms affects the solution, since it seems that adding rooms when the usage is moderate will not affect the solution. Spatial investments consume a considerable part of the education organizations expenses, so avoiding excess room would decrease costs notably. Müller has been able to create schedules when the average usage has been more than 70 percent, so a higher usage rate in itself is not impossible.

An additional sensitivity analysis tool in combination to the optimization could signal about the critical factors for individual solutions. That way todays practice could be questioned more effectively, when the benefits of challenging old practice becomes clear. The barriers to finding better solutions are not always what we think they are, as was noted in this case when it became clear course orders was the largest obstacle to finding better solutions, not the amount of resources available.

The results here considers only one instance, that of house B. Using more instances would make the results more reliable, but it would also have demanded far more time than what has been available for this project. On the other hand, this one instance has been solved several times with different input data, and in all cases, a solution has been found.

If continuing on this research, there are three main directions to go in. The first is to continue with a direct approach, the other to create heuristics for solving the problem. The limits of a direct approach like this have already been noticed during this research. Could a direct approach then be used for the whole campus (without splitting it into parts)? And would the model be enough for the other degree programs? Going in the direction of heuristics we could ask the following questions: How much better results could a heuristic approach produce? Will the improvement be significant enough to cover up for the additional work needed?

A third option would be to enlarge the research to include other fields than university education. Could the model be made even more general, replacing students with customers, degree programs with products and services and the educational activities with any organizational activities? In the literature, most studies are focused on only a certain case, and the results are seldom reproducible or comparable to other researches (Schaerf & Di Gaspero, 2006). The development of more general models and efficient algorithms in addition to them is a rising topic on the research agenda (Schaerf & Di Gaspero 2006; De Causmaecker & Van den Berghe 2012).

5 Conclusions

Scheduling with mixed integer linear programming is possible even for large instances if the problem can be decomposed in sensible parts. In this case study, a schedule was made for the activities in one of the buildings of Hämeenlinnan Ammattikorkeakoulu.

The schedule created was acceptable to the schedulers, and with more accurate information, the created schedule could even be used. The handmade schedules are from some perspectives still better including less free periods. But on the other hand, many of the problems occurring in the scheduling process did not occur in the optimization, for instance there were always enough rooms available since room availability has been considered throughout all the stages. To get a computer model include all the details of human wishes and remarks is not only challenging, it would most probably make the model so complicated it cannot be solved by a direct approach anymore. But it seems like the human schedulers could benefit from using mathematical optimization and then alter the results to ones even better.

Automated scheduling would spare the human schedulers a lot of time, and with the optimization, different constraints hard for the human to perceive could be taken into account. A simple model like the one created here cannot replace human schedulers, but it could be of great help to them. The point of optimization is not to replace people, but to help them do their work better. By using the advantage of information technology, even better schedules can be created.

References

Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operations Research*, *153*(1), 117–135.

De Causmaecker, P., & Van den Berghe, G. (2012). Towards a reference model for timetabling and rostering. *Annals of Operations Research*, *194*(1), 167–176.

Martin, C. (2004). Ohio University's College of Business Uses Integer Programming to Schedule Classes. *Interfaces*, *34*(6), 460–465.

McCollum, B. (2006). A Perspective on Bridging the Gap Between Theory and Practice in University Timetabling. *E.K. Burke and H. Rudová (Eds.): PATAT 2006, Lecture Notes in Computer Science*, *3867*, 3–23.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A., Di Gaspero, L., Qu, R., & Burke, E. (2010). Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. *INFORMS Journal on Computing*, *22*(1), 120–130.

Müller, T. (2009). ITC2007 solver description: a hybrid approach. *BBN Technologies*, *172*(1), 429–446.

Montana, D., Brinn, M., Moore, S., & Bidwell, G. (1998). Genetic Algorithms for Complex, Real-Time Scheduling. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, *3*, 2213–2218.

Murray, K., Müller, T., & Rudová, H. (2006). Modeling and Solution of a Complex University Course Timetabling Problem. *Lecture Notes in Computer Science*, *3867*, 189–209.

Pennanen, A. (2004). *Workplace planning*. Ph.D. thesis, Department of Architecture, Tampere University of Technology, Tampere, Finland. Haahtela-kehitys Oy, Helsinki.

Schaerf, A., & Di Gaspero, L. (2006). Measurability and reproducibility in university timetabling research: Discussion and proposals . *E.K. Burke and H. Rudová (Eds.): PATAT 2006, Lecture Notes in Computer Science*, *3867*, 40–49.

Smith-Miles, K., & Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, *39*(5), 875–889.

Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, *19*(2), 151–162.

Zhai, Y., Sun, S., Wang, J., & Guo, S. (2010). A heuristic algorithm for large-scale job shop scheduling based on operation decompositon using bottleneck machine. 2010 International Conference on Management and Service Science, 1-4.

A Parameters used in the optimization

Table 7: The decision problems difficulty and the parameters and variables used for chosing terms. The size is the number of elements in the vector or matrix.

Variables	3097
Constraints	4096
Non-zero coefficients	74176

Description	**Symbol**	**Size**
Student group	s	17
Term (period)	p	8
Course	a	336
Course credit of course c	$cr(a)$	336
Teacher (Lecturer)	r_L	56
Room type	r_R	17
Maximum time available of room type	$R_{R,MAX}(r_R)$	17
Student group study modules	$StudyModule(a,s)$	5712
Teachers needed	$Resource_L(a,r_L)$	18816
Rooms needed	$Resource_R(a,r_R)$	5712
Course orders	o	1991
Prechosen periods	$ChosenPeriod(a)$	336
Number of years a degree program takes	$years$	4
Number of periods during a year	$periods$	2
Decision variable x	$x(a,p)$	2688
Dummy decision variable y_i	$y_i(p,s)$	136

Table 8: The decision problems difficulty and the parameters and variables used for chosing weeks. The size is the number of elements in the vector or matrix.

Variables	4439
Constraints	11031
Non-zero coefficients	39942

Description	**Symbol**	**Size**
Student group	s	68
Week	p	14
Course	a	181
Total course teaching hours	$TotalHours(a)$	181
Class length of course c	$ClassLength(a)$	181
Minimum week hours	$minH(a,p)$	2534
Maximum week hours	$maxH(a,p)$	2534
Teachers (Lecturers)	r_L	56
Maximum lecture time in a week	$R_{L,MAX}$	25
Room types	r_R	17
Maximum time available of room type	$R_{R,MAX}(r_R)$	17
Student group study modules	$StudyModule(a,s)$	12308
Teachers needed	$CourseTeacher(a,r_L)$	10136
Rooms needed	$Room(a,r_R)$	3077
Average week hours	$awH(s)$	68
Decision variable z	$x(a,p)$	2534
Dummy decision variable y_i	$y_i(p,s)$	952

Table 9: The decision problems difficulty and the parameters and variables used for week 8 and week 11. The amount of courses happens to be the same for both weeks, so the size of the elements in the vectors and matrices are given just once.

Week 8	**Variables**	13951
	Constraints	47319
	Non-zero coefficients	134941
Week 11	**Variables**	13951
	Constraints	47423
	Non-zero coefficients	136115

Description	**Symbol**	**Size**
Student group	s	68
Time slots (hours)	p	45
Days of a week in use	d	5
Hours on a day in use	h	9
Course	a	155
Total course teaching hours	$TotalHours(a)$	155
Class length of course c	$ClassLength(a)$	155
Maximum classes per day	$CPD(a)$	155
Teachers (Lecturers)	r_L	56
Room types	r_R	17
Number of rooms of room type	$R_{R,MAX}(r_R)$	17
Student group study modules	$StudyModule(a, s)$	10540
Teachers needed	$CourseTeacher(a, r_L)$	8680
Rooms needed	$Room(a, r_R)$	2635
Teacher not available	$NoClass(p, r_L)$	2520
Decision variable w	$w(a, p)$	6975
Decision variable y	$y(a, p)$	6975

B The code used for optimization

B.1 Choosing terms for the courses

```
int nCourses = ...;
int nStudentGroup = ...;
int nRoomType = ...;
int nTeachers = ...;
int nPeriod = ...;
int nOrders = ...;
int nP2 = ...;

range Courses = 1..nCourses;
range Periods = 1..nPeriod;
range Teachers = 1..nTeachers;
range StudentG = 1..nStudentGroup;
range RoomType = 1..nRoomType;
range Orders = 1..nOrders;
range P2 = 1..nP2;

float CourseTeachers[Courses][Teachers] = ...;
int StudyModule[Courses][StudentG] = ...;
int Room[Courses][RoomType] = ...;
int CourseCredit[Courses] = ...;
int AvailableRoom[RoomType]= ...;
int OrderBefore[Orders]= ...;
int OrderAfter[Orders]= ...;
int ChosenPeriod[Courses]=...;

dvar boolean x[Courses][Periods];
dvar int y1[Periods][StudentG];
dvar int y2[Periods][StudentG];
dvar int y3[Periods][StudentG];

minimize
  sum(p in Periods)
   sum (s in StudentG)
   (y1[p][s] + 2*y2[p][s] + 0.5*y3[p][s]);

 subject to
{
//All courses must be held
  forall (c in Courses)
    sum (p in Periods)
       x[c][p] == 1;
//Some courses have a prechosen period
  forall (c in Courses)
      if(ChosenPeriod[c]>=1)
   {sum(p in Periods)x[c][p]*(p - ChosenPeriod[c])== 0;}

//The dummy variables for the objective function
  forall (p in Periods)
    forall (s in StudentG)
   {y1[p][s] >= sum (c in Courses)StudyModule[c][s] * x[c][p]
             * CourseCredit[c] - 30;
      y1[p][s] >= 0;
      y2[p][s] >= sum (c in Courses)StudyModule[c][s] * x[c][p]
             * CourseCredit[c] - 31;
      y2[p][s] >= 0;
      y3[p][s] >= 28 - sum (c in Courses)StudyModule[c][s]
             * x[c][p] * CourseCredit[c];
      y3[p][s] >= 0;}

//Room constraint
  forall (i in P2)
    forall (r in RoomType)
      sum (c in Courses)
       (x[c][i] * Room[c][r] + x[c][i+2] * Room[c][r] + x[c][i+4] * Room[c][r]
         + x[c][i+6] * Room[c][r]) <= 0.8 * AvailableRoom[r];

//Teacher constraint
  forall (i in P2)
    forall (t in Teachers)
      sum (c in Courses)
       (x[c][i] * CourseTeachers[c][t]
        + x[c][i+2] * CourseTeachers[c][t]
        + x[c][i+4] * CourseTeachers[c][t]
        + x[c][i+6] * CourseTeachers[c][t])<= 25;

//Courses in a particular order
      forall (c in Orders)
     { (sum (p1 in Periods) p1 * x[OrderBefore[c]][p1] + 1 )
       <= sum (p2 in Periods) p2 * x[OrderAfter[c]][p2];}
}
```

B.2 Weekly lecture allocations

```
int nCourses = ...;
int nStudentGroup = ...;
int nRoomType = ...;
int nTeachers = ...;
int nWeeks = ...;

range Courses = 1..nCourses;
range Weeks = 1..nWeeks;
range Teachers = 1..nTeachers;
range StudentG = 1..nStudentGroup;
range RoomType = 1..nRoomType;

float ClassTeachers[Courses][Teachers] = ...;
int StudyModule[Courses][StudentG] = ...;
float Room[Courses][RoomType] = ...;
int AvailableRoom[RoomType]= ...;
int CourseLessons[Courses]= ...;
int MinHours[Courses][Weeks]= ...;
int MaxHours[Courses][Weeks]= ...;
int AverageWeekHours[StudentG]= ...;
int ClassLength[Courses]=...;

dvar int z[Courses][Weeks];
dvar int y1[StudentG][Weeks];
dvar int y2[StudentG][Weeks];

minimize

      sum (s in StudentG)
         sum (n in Weeks)
      (y1[s][n]+ 2*y2[s][n]);


 subject to
{
//All lessons must be held
  forall (c in Courses)
    sum (n in Weeks)
       z[c][n] == CourseLessons[c]/ ClassLength[c];

//Minimum amout of lessons each week
  forall (c in Courses)
    forall (n in Weeks)
       z[c][n] >= MinHours[c][n] / ClassLength[c];

//Maximum amout of lessons each week
  forall (c in Courses)
    forall (n in Weeks)
       z[c][n] <= MaxHours[c][n]/ ClassLength[c];
//Maximum amout of lesson hours each week
 forall (n in Weeks)
    forall (s in StudentG)
      sum (c in Courses)
       z[c][n] * ClassLength[c] * StudyModule[c][s] <= 45;

//Dummy variables for the objective function
  forall (n in Weeks)
    forall (s in StudentG)
  {y1[s][n]>= sum (c in Courses)StudyModule[c][s] *  z[c][n]  * ClassLength[c]
             - AverageWeekHours[s];
   y1[s][n]>= 0;
   y2[s][n]>= sum (c in Courses)StudyModule[c][s] *  z[c][n]  * ClassLength[c]
             - AverageWeekHours[s] - 5;
   y2[s][n]>= 0;}

// Room constraint
  forall (n in Weeks)
    forall (r in RoomType)
      sum (c in Courses)
       z[c][n] * ClassLength[c] * Room[c][r] <= 0.9 * AvailableRoom[r];

// Teacher constraint
 forall (n in Weeks)
    forall (l in Teachers)
      sum (c in Courses)
       z[c][n] * ClassLength[c] * ClassTeachers[c][l] <= 30;
}
```

B.3 Creating the schedule

```
int nClasses = ...;
int nStudentGroup = ...;
int nRoomType = ...;
int nTeachers = ...;
int nTime = ...;
int nTime44 =...;
int nTime43 =...;
int nTime37 =...;
int nDayTime =...;
int nDayTime8 =...;
int nDay =...;

range Classes = 1..nClasses;
range Time = 1..nTime;
range Teachers = 1..nTeachers;
range StudentG = 1..nStudentGroup;
range RoomType = 1..nRoomType;
range Time44 = 1..nTime44;
range Time43 = 1..nTime43;
range Time37 = 1..nTime37;
range DayTime = 1..nDayTime;
range DayTime8 = 1..nDayTime8;
range Day = 1..nDay;

int ClassTeachers[Classes][Teachers] = ...;
int StudyModule[Classes][StudentG] = ...;
int Room[Classes][RoomType] = ...;
int AvailableRoom[RoomType]= ...;
int ClassLectures[Classes]=...;
int ClassLength[Classes]=...;
int ClassesPerDay[Classes]=...;
int NoClass[Time][Teachers]=...;
int Mu[DayTime]=...;

dvar boolean y[Classes][Time];
dvar boolean w[Classes][Time];

minimize

  sum (c in Classes)
   sum (d in Day)
     sum (h in DayTime)
       Mu[h]*(w[c][h+(d-1)*9] + y[c][h+(d-1)*9]);

 subject to
{
//Every class must be held
  forall (c in Classes)
    sum (t in Time)
       w[c][t] == ClassLectures[c] / ClassLength[c];

//class hours must equal total class lectures
  forall (c in Classes)
    sum (t in Time)
       (w[c][t]+y[c][t]) == ClassLectures[c];

//class length = 1, no extra constraints

//class length = 2
```

```
  forall (c in Classes)
    if(ClassLength[c]==2)
    {w[c][45] == 0;
     w[c][36] == 0;
     w[c][27] == 0;
     w[c][18] == 0;
     w[c][9] == 0;}

    forall (c in Classes)
      forall (t in Time44)
     if(ClassLength[c]==2)
     w[c][t] == y[c][t+1] ;

//class length = 3

  forall (c in Classes)
    if(ClassLength[c]==3)
    {w[c][45] == 0;
     w[c][44] == 0;
     w[c][36] == 0;
     w[c][35] == 0;
     w[c][27] == 0;
     w[c][26] == 0;
     w[c][18] == 0;
     w[c][17] == 0;
     w[c][8] == 0;
     w[c][9] == 0;}

  forall (c in Classes)
      forall (t in Time43)
        if(ClassLength[c]==3)
       {w[c][t] <=y[c][t+1];
       w[c][t]<=y[c][t+2];}

//class length = 8
  forall (c in Classes)
    forall (d in Day)
      forall (t in DayTime8)
     if(ClassLength[c]==8)
     {w[c][1 + t + (d-1)*9]==0;}

  forall (c in Classes)
      forall (t in Time37)
        if(ClassLength[c]==8)
       {w[c][t]<=y[c][t+1];
       w[c][t]<=y[c][t+2];
       w[c][t]<=y[c][t+3];
       w[c][t]<=y[c][t+4];
       w[c][t]<=y[c][t+5];
       w[c][t]<=y[c][t+6];
       w[c][t]<=y[c][t+7];}

//other contraints

//only one classlecture at a time
  forall (t in Time)
      forall (c in Classes)
    (w[c][t]+y[c][t]) <= 1;

// allowed number of classes each day
  forall (c in Classes)
    forall (d in Day)
      sum (t in DayTime)
    w[c][t + (d-1)*9] <= ClassesPerDay[c];

// room constraint
  forall (t in Time)
    forall (r in RoomType)
      sum (c in Classes)
    (w[c][t]+y[c][t]) * Room[c][r] <=AvailableRoom[r];

// teacher constraint
  forall (t in Time)
    forall (l in Teachers)
      if(NoClass[t][l]==1)
      {sum (c in Classes)
       (w[c][t]+y[c][t]) * ClassTeachers[c][l] ==0;}
      else
      {sum (c in Classes)
       (w[c][t]+y[c][t]) * ClassTeachers[c][l] <= 1;}

// student group constraint
   forall (t in Time)
      forall (s in StudentG)
         sum (c in Classes)
             StudyModule[c][s] * (w[c][t]+y[c][t]) <= 1;
}
```