CHALMERS | GÖTEBORGS UNIVERSITET

Modeling and optimization of university timetabling

A case study in integer programming

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Johan Havås
Alfred Olsson
Jim Persson
Mirjam Sophia Schierscher

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2013

Modeling and optimization of university timetabling

A case study in integer programming

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Mirjam Sophia Schierscher

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk fysik vid Chalmers
Jim Persson

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers
Johan Havås Alfred Olsson

Handledare: Zuzana Šabartová
Examinator: Hjalmar Rosengren

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2013

Abstract

Timetabling is a task that has to be resolved at any school or university. The fact that it is such a common problem and that it is often a large and complex issue makes it an interesting and suitable subject for mathematical optimization. This report presents a model for the problem of scheduling a number of courses given by the department of Mathematical Sciences of the University of Gothenburg and Chalmers University of Technology. It is modeled as an integer programming problem where the constraints take into account the requirements that are necessary for the timetable to be valid and the objective function is chosen in such a way that it reflects the preferences of students and teachers. The model is subsequently solved using AMPL with the solver CPLEX and the results are visualized so that they may provide guidance on areas where the current timetable may be improved. Sensitivity analysis is performed in order to investigate how the solution is affected when various conditions are changed.

Sammanfattning

Schemaläggning är en uppgift som måste hanteras på varje skola och universitet. Att det är ett så vanligt problem och att det ofta är en stor och komplex fråga gör det till ett intressant och lämpligt ämne för matematisk optimering. Denna rapport presenterar en modell för problemet att schemalägga ett antal kurser som ges av Institutionen för Matematiska vetenskaper vid Göteborgs Universitet och Chalmers tekniska högskola. Detta modelleras som ett heltalsoptimeringsproblem där bivillkoren tar hänsyn till de krav som är nödvändiga för att schemat ska vara praktiskt möjligt och målfunktionen väljs på ett sådant sätt att den återspelgar studenters och lärares preferenser. Modellen löses sedan i AMPL med lösaren CPLEX och resultaten visualiseras så att de kan ge vägledning på områden där det nuvarande schemat kan förbättras. Känslighetsanalys genomförs för att undersöka hur lösningen påverkas när diverse förutsättningar ändras.

Preface

This section presents how work and report writing have been divided within the group. In order to keep track of all group members' contribution, a logbook and time log was kept during the project. These were used as base when writing this section.

Working process

To efficiently contribute to the project, a complete overview has been essential and hence there has been no division of responsibility within the group. There has been at least one group meeting every week. At these meetings, the current status of the project has been discussed as well as planning for upcoming weeks. All big decisions were taken on these meetings.

Most of the work with the model and implementation in AMPL was performed in group, and most of the writing was made individually. The group has had constant contact via e-mail, and everything that was done was put into a shared folder on Google Drive so that every group member had access to everything at all times.

Although everyone has been involved in all parts of the project, some members of the group were more involved in some parts. Jim was more responsible for the report and Mirjam, Alfred and Johan spent more time with the model and implementing in AMPL. The sensitivity analysis was made by Alfred and Jim, and extending the model to the second study period was done by Mirjam and Johan.

Report

Here the authors of the different sections of the report are presented. Although not referenced, every author has contributed with suggestions to all sections.

Johan Havås: Integer programming, Branch-and-bound, Conclusions.

Alfred Olsson: Cutting-plane methods, AMPL and CPLEX, Discussion, Conclusions.

Jim Persson: Introduction, Modeling and mathematical programming, Convexity, Linear programming, Results, Sensitivity analysis, Conclusions.

Mirjam Sophia Schierscher: Problem description, Mathematical formulation, Results, Conclusions.

Acknowledgements

We would like to thank our supervisor Zuzana Šabartová for her support and encouragement during this project. We are also grateful to Jeanette Montell-Westerlin and Ulla Dinger for their help in collecting data and their advice on how to make a good timetable for as many as possible.

Contents

1 Introduction

Optimization is a branch of mathematics concerned with finding the best solution to a given problem. This is done by simplifying the problem and expressing it in mathematical notation to create a so-called mathematical model describing the problem. There are many areas where optimization is frequently used, for example in production where the goal might be to minimize the costs of manufacturing a certain product, or when trying to find the most efficient route for deliveries to certain destinations.

A typical optimization problem faced in a wide range of areas today is scheduling, for example timetabling of courses at universities or scheduling of airline flights. The goal is to find the best possible schedule that is not only practically possible, but also meets the preferences of all parties involved. Specific applications of scheduling can be found in Pinedos work [1].

Even though optimization is an effective technique for finding the best solution to a variety of problems, some problems are very large and thus finding a solution can be very time consuming. However, with increasing technical resources, many problems are today manageable. For problems that are linear there exists a powerful solution algorithm called the simplex method, which will be discussed further in Section 2.3. This method was developed by George Dantzig, who was one of the pioneers within optimization theory. A detailed description of the algorithm is written by Dantzig G. and Thapa M. N., in [2].

1.1 Background

This report describes how to solve the timetabling problem for a number of courses at the department of Mathematical Sciences of the University of Gothenburg and Chalmers University of Technology through the use of integer programming.

Integer programming is a common approach to solve this kind of problem. It has been applied by, among others, Daskalaki S. et al. in [3], and Dimopoulou M. and Miliotis P. in [4]. Another possible approach is to make use of metaheuristic optimization, which is a new, more complicated area of optimization. This was studied by Lewis R. in [5].

The timetabling at the mathematics department is currently accomplished by reusing the timetable from previous year as closely as possible and manually correcting upcoming problems. This is very time consuming and may take several weeks to finish, whereas an automatized computational method would make the process faster and less demanding. Such a method can be obtained by formulating a mathematical model of the given timetabling problem, with respect to number of students, courses and rooms.

1.2 Specification of timetabling problem

When constructing a timetable, considerations must be taken to underlying rules telling whether two events may occur at the same time. This implicates that there are some properties of a timetable that must be fulfilled. The problem lies in placing just one teacher and one class, in one classroom, at one time, without any collisions. Coincidently, there is a required number of lecture hours per course that must be scheduled every week. Besides conditions that must be fulfilled, there are also personal opinions of teachers and students about what makes a good schedule. This will be discussed in detail in Section 3.

This case study is made for the spring of 2013, including two study periods, each of which has a length of eight weeks. In the first study period there are 15 courses and in the second study period there are 13 courses. There are 11 rooms, and the number of students for the courses varies between 4 and 68. In this case study, everything can be modeled without taking teachers into account, since every course has one assigned teacher who is supposed to be available at all times. Notice that this is how it is currently done. Even so, teachers may have some preferences which will have to be taken into account.

The resulting timetable will be one that is suitable for the mathematics department, but it may collide with other departments' timetables and may therefore not be a valid schedule in practice. This is because all departments make a preliminary, potential first version of their timetable. Then different departments compare and discuss their suggestions in order to see if anything collides, e.g., courses in one program that are possible to take for students at another program may not collide with other courses from either of these two programs. If collisions are found, changes must be made. We do not have the possibility to discuss the resulting timetable with other departments. However, if some last changes were to be made, this would not be a problem with a mathematical tool doing all the work.

In Figure 1, the current timetable for the first study period of spring 2013 is shown. Notice that there are some courses that are scheduled in the rooms EF and VF, which do not belong to the mathematics department. This is due to the difficulty of finding appropriate rooms for these courses, seeing as the number of students taking them is quite high. However, when the timetabling is made with integer programming, this is not an issue.

1.3 Purpose

The purpose of this report is to, through the use of integer programming, determine a valid timetable for courses given by the department of Mathematical Sciences at the University of Gothenburg and Chalmers University of Technology during the spring of 2013. The timetable must fulfill all requirements necessary to be practically possible and it should be optimized with respect to preferences from teachers and students. The formulated model should be general enough that it may be used for other study periods and departments by making small adjustments.

1.4 Delimitations

There will be two delimitations. One is that it will be assumed that every teacher is available at all times. In other words, there will be no consideration taken to if a teacher has another course in another program at the same time. This is actually the way the timetable is done today, hence this will not be a practical problem. However, this delimitation makes the model slightly less general. The second delimitation is that the three available computer rooms were modeled as one larger room. This simplification was made because it is common that all computer rooms are reserved for a single lab session.

1.5 Method

To aquire the required understanding of optimization and how to construct the mathematical model, some literature studies were needed. Since all group members do not have the same mathematical background, all literature studies were made individually.

Information about the courses and teachers was collected from the people responsible for timetabling at the mathematics department. They also provided information on what times teachers generally prefer. Additional information was found on the courses' homepages.

A model was formulated and implemented in the computer language AMPL and solved with the solver CPLEX, described in Section 2.5. MATLAB was used to visualize the resulting timetable, which allowed comparison with the timetable that is currently in use. The model was also examined with sensitivity analysis, where the stability was studied as well as the influence of different components of the model.

	Monday		Tuesday		Wednesday		Thursday		Friday	
8:00	MMG800 lec	Pascal	MMG800 lec	Pascal	MMG720 lec	MVH12	MMGL31 ex	MVF26	MMA421 lec	MVF21
	MMGL31 ex	MVF26	LGMA10 ex	MVF31			MMG720 ex	MVH12	MMG800 lec	Pascal
									LGMA10 ex	MVF31
10:00	MMG500 lec	EF	MMGK11 lec	Euler	MMGF20 lec	Pascal	MMG500 lec	EF	MMGK11 lec	Euler
	MMGK11 lec	Euler	MMGF30 lec	MVF23	MMA421 ex	MVF21	MMGK11 lec	Euler	MMGF30 lec	MVF23
	MMA511 lec	MVF23	MMGL31 lec	MVF33	LGMA10 ex	MVF23	MMGL31 lec	MVF26	MMA511 lec	MVF33
	MMGL31 lec	MVF26	MMG300 lec	Pascal	MSG830 ex	MVF33	MMG300 lec	MVH12	MMGL31 lec	Pascal
	MMG720 lec	MVH12	MSG830 lec	VF	MMGL31 com	MVF22	MMG800 lec	Pascal	MMGF20 ex	MVF26
	MMGF20 lec	Pascal	LGMA10 ex	MVF31			LGMA10 ex	MVF31	LGMA10 ex	MVF31
	LGMA10 ex	MVF31	MMG500 ex	MVF32					MMG500 ex	MVF32
			MMG500 ex	MVH12					MMG500 ex	MVH12
13:15	LGMA10 lec	EF	LGMA10 lec	Euler	MVG300 lec	Euler	MSG830 lec	Euler	LGMA10 lec	EF
	MVG300 lec	Euler	MSA200 lec	MVF23	MMGF30 lec	MVF23	MMA511 lec	MVF23	MVG300 lec	Euler
	MMA421 lec	MVF21	MMG300 ex	MVF26	MSG200 lec	Pascal	LGMA10 lec	Pascal	MMGK11 ex	MVF21
	MSG200 lec	Pascal	MMGK11 ex	MVF31	LGMA10 ex	MVF21	MMGL31 ex	MVF21	MSA200 ex	MVF23
	MMGK11 ex	MVF23	MMGL31 ex	MVF33	MMGL31 com	MVF22	MMGK11 ex	MVF31	MMGK11 ex	MVF31
	MMGL31 ex	MVF26	MMGK11 ex	MVH11			MMGK11 ex	MVH11	MMGL31 ex	MVF33
	MMGK11 ex	MVF31	MMGF20 ex	MVH12			MMG300 ex	MVH12	MSG200 ex	MVH12
			MSG830 com	MVF22					MSG830 com	MVF22
15:15	MVG300 com	MVF22	MSG200 ex	MVH12	MSA200 lec	MVF23			MVG300 com	MVF22
					MVG300 com	MVF22				

Figure 1: The current timetable for all courses in the first study period of spring 2013. As seen, a week is divided into five days where every day is divided into four time periods. Notice that all sessions are presented with the course code and type of session to the left and the room to the right. Notice that the rooms EF and VF are not part of the mathematics department.

2 Theory

The analysis of the given optimization problem and related algorithms requires understanding of the key concepts in optimization theory. Therefore, the essential ideas and basic facts used to formulate and solve the described timetabling problem are presented.

2.1 Modeling and mathematical programming

The general idea of optimization, also referred to as mathematical programming, is to find the best solution to a given problem. This is done by employing different optimization algorithms to the mathematical description of the problem, which consists of an *objective function* $f(\mathbf{x})$, that is to be minimized or maximized, and a set of *constraints*. These constraints are of two different types, so-called *hard constraints*, which have to be fulfilled, and *soft constraints* which should be fulfilled if possible. The objective function expresses the objective of the given problem in terms of variables that can be controlled, so-called *decision variables* $\mathbf{x}$. The variables that satisfy the given constraints define the *feasible set*. The objective function, together with the set of constraints, is what builds the mathematical model for the given problem.

Definition 1. *A general mathematical programming problem is defined as*

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}), \\ &\text{s.t.} \quad \mathbf{x} \in X, \end{aligned}$$

where s.t. *is an abbreviation for 'subject to'. Here, the objective function is $f(\mathbf{x}) : X \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$, where $n \in \mathbb{N}$, is the feasible set determined by constraints.*

The problem here is written as a minimization problem, which is usually done by convention. A maximization problem can easily be rewritten as a minimization problem by minimizing the negative objective function. The algorithms for solving many optimization problems require considerable numerical effort but more efficient methods are available for certain types of problems. Examples of such problems are linear programs and integer programs, which can be used to model many real world problems. A linear programming (LP) problem consists of only linear functions describing the objective function and the set of constraints. Linear programs will be discussed more deeply in Section 2.3. A problem where all decision variables are integers and the objective function and the constraints are conventionally linear is called an integer programming (IP) problem and is described further in Section 2.4. The problem solved in this report is an integer programming problem and therefore we mainly focus on this type of problems and the algorithms for solving them. There are some other types of mathematical programs, for example non-linear programs and stochastic programs, but these will not be discussed in this report. Non-linear programming problems are more difficult to solve than LP problems, since there is no single method that can solve a general non-linear model. For LP problems there do exist general methods, where the most used is *the simplex method* which will be discussed in Section 2.3. An additional important term in optimization, that benefits the solution methods, is *convexity*.

2.2 Convexity

A problem is convex if the objective function is a convex function and the feasible set is a convex set.

Definition 2. *A set S is said to be convex if it satisfies*

$$\left.\begin{aligned}\mathbf{x}^1, \mathbf{x}^2 \in S\\ \lambda \in (0,1)\end{aligned}\right\} \Rightarrow \lambda\mathbf{x}^1 + (1-\lambda)\mathbf{x}^2 \in S. \tag{1}$$

To get an intuitive understanding of what this means, consider the two-dimensional case. A two dimensional set is convex if it is possible to draw a straight line between any two points in the set, without the line ever leaving the set.

Definition 3. *A function $f(\mathbf{x}) : X \to \mathbb{R}$ is said to be convex on the convex feasible region X if for all points $\mathbf{x}^1, \mathbf{x}^2 \in X$ and $\lambda \in (0,1)$ we have that*

$$f\left(\lambda\mathbf{x}^1 + (1-\lambda)\mathbf{x}^2\right) \leq \lambda f(\mathbf{x}^1) + (1-\lambda)f(\mathbf{x}^2).$$

In other words, a two dimensional function is convex if the function values between two arbitrary points on the graph of the function lie below the straight line between these two points.

The optimal solution to a problem is given as a global minimum of the objective function on the feasible set. Solution methods are often designed to find local minima, even though the solution of interest is the global minimum. For convex problems it is a fact that a local minimum is also a global minimum [6]. This makes the convexity property useful in optimization, since there are not many effecient methods to solve problems that are non-convex.

2.3 Linear programming

In linear programming both the objective function and the constraints are linear. A general linear programming problem is a problem of the form

$$\begin{aligned}
\min z = &\sum_{j=1}^{n} c_j x_j, \\
\text{s.t.} \qquad &\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, \ \ldots, m, \\
&\mathrm{x}_j \geq 0, \qquad j = 1, \ \ldots, n.
\end{aligned} \tag{2}$$

Here, c_j is the weight coefficient of the objective function at variable x_j and a_{ij} and b_i are the coefficients of the constraints. A linear program is obviously always convex, for a detailed proof see [6].

A common approach for solving LP problems is to use the simplex method, which is often an efficient method. This method is designed for linear programs in standard form, that is, the problem should be rewritten to consist of only positive variables and the inequalities should be replaced by equalities. Changing an inequality to an equality is possible by introducing new variables, so-called *slack variables.* To clarify this step we look at an example.

Consider the following inequality:

$$x_1 + x_2 \geq b, \qquad x_1,\, x_2 > 0.$$

It is possible to find a variable s_1 such that

$$x_1 + x_2 + s_1 = b,$$

where $s_1 \geq 0$ is a slack variable. A free of sign variable x_a can be rewritten by making the substitution

$$x_a = x_a^+ - x_a^-,$$

where $x_a^+, x_a^- \geq 0$, see [6]. The problem (2) can now be rewritten in standard form as

$$\begin{aligned} \min z = & \sum_{j=1}^{n} c_j x_j, \\ \text{s.t.} \qquad & \sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1,\ \ldots,\ m, \\ & x_j \geq 0, \qquad j = 1,\ \ldots,\ n+k, \end{aligned} \tag{3}$$

where x_j, $j = n+1, \ldots, n+k$, are slack variables. The inequality sign has been changed to an equality sign by appropriate choices of the slack variables.

It is now possible to define a linear problem in standard form in the more compact matrix form.

Definition 4. *A linear programming problem is said to be in the standard matrix form if it is expressed as*

$$\begin{aligned} \min z = & \ \mathbf{c}^T \mathbf{x}, \\ \text{s.t.} \qquad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq 0, \end{aligned} \tag{4}$$

where $A \in \mathbb{R}^{m \times n}$, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \geq \mathbf{0}^m$. Here, $\mathbf{c}^T$ is the transpose of $\mathbf{c}$.

The simplex method

The simplex method utilizes the fundamental theorem of linear programming, which says that the optimal solution to a problem is found in an *extreme point* of the feasible set of solutions. To prove this theorem, we first need to define the term extreme point.

Definition 5. *A point $\mathbf{v}$ of a convex set S is called an extreme point if whenever*

$$\mathbf{v} = \lambda \mathbf{x}^1 + (1-\lambda)\mathbf{x}^2,$$

where $\mathbf{x}^1, \mathbf{x}^2 \in S$ and $\lambda \in (0,1)$, then

$$\mathbf{v} = \mathbf{x}^1 = \mathbf{x}^2.$$

For a set in two or three dimensions, an extreme point can be thought of as a corner.

Theorem 1: Fundamental theorem of linear programming.
Assume that the feasible region X of an LP problem is bounded and non-empty, i.e., there exists a bounded optimal solution to the LP problem. Then the minimum value of the objective function $\mathbf{c}^T\mathbf{x}$ is obtained in an extreme point $\mathbf{x}_k$ of X.

The proof can now be presented as in *Optimization* by Lundgren et al. [6].

Proof. Assume the opposite, i.e., there exists an inner point or a boundary point (not an extreme point) $\hat{\mathbf{x}}$ that minimizes $\mathbf{c}^T\mathbf{x}$, and no extreme point $\mathbf{x}_k$ to X gives the same objective function value, i.e., $\mathbf{c}^T\hat{\mathbf{x}} < \mathbf{c}^T\mathbf{x}_k$, $k = 1, \ \ldots, \ p$. Since the feasible region defines a convex set, every feasible point can be written as a convex combination of the extreme points in X. Hence also the point $\hat{\mathbf{x}}$, i.e., $\hat{\mathbf{x}}$ can be expressed as

$$\hat{\mathbf{x}} = \sum_{k=1}^{p} \lambda_k \mathbf{x}_k,$$

where

$$\sum_{k=1}^{p} \lambda_k = 1, \qquad \lambda_k \geq 0, \ k = 1, \ \ldots, \ p.$$

This implies that

$$\mathbf{c}^T\hat{\mathbf{x}} = \mathbf{c}^T \sum_{k=1}^{p} \lambda_k \mathbf{x}_k = \sum_{k=1}^{p} \lambda_k \mathbf{c}^T\mathbf{x}_k > \sum_{k=1}^{p} \lambda_k \mathbf{c}^T\hat{\mathbf{x}} = \mathbf{c}^T\hat{\mathbf{x}} \sum_{k=1}^{p} \lambda_k = \mathbf{c}^T\hat{\mathbf{x}},$$

which is a contradiction. Therefore, the assumption that there exists an inner point or a boundary point that minimizes $\mathbf{c}^T\mathbf{x}$ is false. □

What the simplex method does is to search through the extreme points to find which extreme point gives the optimal solution [2]. This is done by systematically examining one solution after another, in a direction where the value of the objective function is improved. Linear problems are convex problems, hence a found local minimum also qualifies as a global minimum.

Before explaining the simplex algorithm, we need the definition of a basic feasible solution.

Definition 6. *For a problem in standard form, with the additional properties that $A \in \mathbb{R}^{m \times n}$ and $\operatorname{rank} A = \operatorname{rank}(A,\mathbf{b}) = m$, $n > m$, a point $\tilde{\mathbf{x}}$ is a basic solution if:*

1. *The equality constraints are satisfied at $\tilde{\mathbf{x}}$, that is $A\tilde{\mathbf{x}} = \mathbf{b}$.*

2. *The columns of A corresponding to the non-zero components of $\tilde{\mathbf{x}}$ are linearly independent.*

A basic solution that also satisfies the non-negativity constraints $\mathbf{x} \geq \mathbf{0^n}$ and $B^{-1}\mathbf{b} \geq \mathbf{0^m}$ is called a basic feasible solution.

We also need to present and explain some notation. The decision variables $\mathbf{x}$ can be divided into basic and non-basic variables according to $\mathbf{x} = (\mathbf{x}_B^T, \mathbf{x}_N^T)^T$. Here, $\mathbf{x}_B$ and $\mathbf{x}_N$ denotes basic and non-basic variables respectively. Non-basic variables are variables that are set to zero in order to find a basic solution to the problem, and the basic variables are the variables that are not set to zero. In the same way as the decision variables can be divided into two parts, so can the cost coefficients $\mathbf{c} = (\mathbf{c}_B^T, \mathbf{c}_N^T)^T$ and the matrix $A = (B|\ N)$. Here, B consists of the coefficients of the basic variables and N consists of the coefficients of the non-basic variables.

It is possible to prove that a basic feasible solution is equivalent to an extreme point, see [7]. The simplex algorithm starts from a point, chosen as a basic feasible solution to the problem. If there is no known basic feasible solution, the algorithm starts with finding such a point. The problem can then be divided into two parts, phase I where we find a starting point and phase II where the solution to the given problem is found. First, phase II is described, assuming a basic feasible solution to the given problem is known. The simplex algorithm can now be described in a step by step procedure, following the description in *An Introduction to Continuous Optimization* by Andréasson M. et al. [7]:

1. Choose the basic feasible solution $\mathbf{x} = (\mathbf{x}_B^T, \mathbf{x}_N^T)^T$ as a starting point.
2. Determine which direction will be most beneficial to the solution, by calculating reduced costs of the non-basic variables:
$$(\tilde{\mathbf{c}}_N)_j := (\mathbf{c}_N^T - \mathbf{c}_N^T B^{-1} N)_j, \qquad j = 1, \ldots, n-m, \tag{5}$$
where $(\tilde{\mathbf{c}}_N)_j$ is a vector of the reduced costs. For the j that minimizes $(\tilde{\mathbf{c}}_N)_j$ we choose $(\mathbf{x}_N)_j$ to enter the basis. If $(\tilde{\mathbf{c}}_N)_j \geq 0$ for all $j = 1, \ldots, n-m$, then no direction is beneficial and the optimal solution is found.
3. Determine the leaving variable, equivalent to find how long it is possible to change the ingoing variable in the chosen direction, without leaving the feasible region. This is done by choosing $(\mathbf{x}_B)_i$ for the i that minimizes
$$\frac{(B^{-1}\mathbf{b})_i}{(B^{-1}N_j)_i}, \qquad i \in \{k | (B^{-1}N_j)_k > 0\}. \tag{6}$$
If
$$B^{-1}N_j \leq \mathbf{0}^m,$$
then the problem is unbounded and the algorithm stops.
4. Let $(\mathbf{x}_N)_j$ take the place of $(\mathbf{x}_B)_i$ in the basis and repeat from step 1.

To find a possible starting point and solve phase I, consider the situation where a solution to the problem (4) is wanted. By introducing so-called *artificial variables* $\mathbf{a} \in \mathbb{R}^m$ it is possible to formulate the new problem:

$$\begin{aligned} \min w = (\mathbf{1}^m)^T \mathbf{a}&, \\ \text{s.t} \qquad A\mathbf{x} + \mathbf{I}^m \mathbf{a} &= \mathbf{b}, \\ \mathbf{x} &\geq 0, \\ \mathbf{a} &\geq 0. \end{aligned} \tag{7}$$

An additional variable a_i, $i = 1, \ldots, m$ is introduced for every linear constraint. If $\mathbf{a}$ is taken as the basic variable and $\mathbf{x}$ as the non-basic variable, then B will be the unit matrix in $\mathbb{R}^{m \times m}$. This will in fact be a basic feasible solution to problem (7) and can be used as a starting point for the simplex method. This new problem can now be solved by the algorithm as described above. It is possible to show, see [7], that for a given optimal solution to w, we must have that $\mathbf{a} = \mathbf{0}$ and that $\mathbf{x}$ is a basic feasible solution to problem (4). If the solution to w instead is greater than zero, it means that the original problem is infeasible since $A\mathbf{x} + \mathbf{I}^m\mathbf{a} = \mathbf{b}$ could not be fulfilled with $\mathbf{a} = \mathbf{0}$, and this was required in the original problem.

2.4 Integer programming

In many real world problems some variables may not take any value. For example if the question is whether a specific project should be carried through or not we would not get any useful information with a linear model. We would need a variable that takes the value one if the project should be realized and zero otherwise. Then we get an IP problem, i.e., when all variables may only take integer values. Solving it straightforwardly as an LP problem and rounding the LP solution does not necassarily give the optimal solution. If the variable in the above example can take any value between zero and one, we can not conclude that the project should be realized even if we get 0.99 as the optimal solution. There could be several other aspects that make this choice disadvantageous, for instance if there is not enough money available to complete the project and it is of no use when it is incomplete.

Other examples of IP problems are scheduling problems, assignment problems, and *the traveling salesman problem*. Application of IP to these topics can be found in [8], [9], and [10]. A general IP problem may look like

$$\begin{aligned} \min z = &\sum_{j=1}^{n} c_j x_j, \\ \text{s.t.} \quad &\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m, \\ &x_j \geq 0 \ \text{ integer}, \qquad j = 1, \ldots, n. \end{aligned}$$

IP problems are generally much harder to solve than LP problems. This is because of the fact that the optimal solution can be anywhere in the discrete set of possible solutions, not only in extreme points. This set is non-convex, i.e., if we take any two points in this set and connect them with a straight line it will always pass through infinitely many points lying outside the feasible set. The easiest way to solve an IP problem is to compute all possible solutions and simply choose the best one. This obvious method works in theory for all integer problems, but it takes too much time in practice for large problems. Therefore several other teqniques have been developed. The most used are variants of the *branch-and-bound* method, usually combined with a *cutting-plane* method, then called *branch-and-cut*. Other used methods are *heuristics*, i.e., non-optimizing methods. Heuristics usually find a feasible solution relatively quickly but do not say anything about how close to the optimum it is [11]. Sometimes heuristics are combined with algorithms that try to improve that solution.

The branch-and-bound method and cutting-plane methods will be discussed and explained further. For this we need the definition of *LP relaxation* for a binary problem.

Definition 7. *The LP relaxation of a binary IP problem is the problem that arises by replacing the constraint that each variable must be 0 or 1 by the weaker constraint that each variable belongs to the closed interval* [0,1]. *That is, each constraint of the form*

$$x_i \in \{0,1\}$$

is replaced by the two constraints

$$0 \leq x_i \leq 1.$$

Branch-and-bound

The idea of the branch-and-bound method is to divide the problem into subproblems and solve LP relaxed versions of these problems, this is the branching step. The subproblems can be defined by fixing a subset of the variables either to 0 or 1, for further details see [6]. For each subproblem we try to estimate how good a solution to this problem is and this gives a bound to the optimal solution. The number of problems will increase, but on the other hand LP problems are easy to solve, e.g., with the simplex method, as described in Section 2.3. Also, if there is no solution to a relaxed problem, there is no solution to the original problem.

To demonstrate the method we look at a simple example. Assume that our problem is

$$\begin{aligned} &\min z = x_1 + 2x_2 + 3x_3, \\ &\text{s.t.} \qquad 2x_1 + 2x_2 + 2x_3 \geq 3, \\ &\qquad\qquad\qquad x_1,\, x_2,\, x_3 \in \{0,\, 1\}. \end{aligned}$$

The procedure of solving this problem with branch-and-bound is illustrated in Figure 2. We start by solving the LP relaxation to the original problem. The optimal objective value z_0 of the relaxed problem P_0 is a lower bound to the objective function in our original problem, i.e., there is no feasible solution with a better value. This follows from the fact that the set containing the solutions to the relaxed problem also contains all solutions to the original problem. We take P_0 as the starting node in our tree. The solution to P_0 is not feasible and therefore we divide the feasible region into two parts, by fixing $x_2 = 0$ and $x_2 = 1$. We start with $x_2 = 1$ and denote this subproblem P_1. We continue in the same fashion until we find our first feasible solution in P_2. This solution to the LP relaxation have all variables binary and is an upper bound to the original problem. When we solve the LP relaxed problem in P_3 we see that the solution is worse than our upper bound, therefore we do not have to explore this branch further. Now we have examined every possibility with $x_2 = 1$, thus we continue with the branch where $x_2 = 0$. If we find a lower feasible solution than the current best, the upper bound of the optimal solution is updated. The search continues until each branch is either computed or neglected. When the search is finished we have the optimal solution in our upper bound.

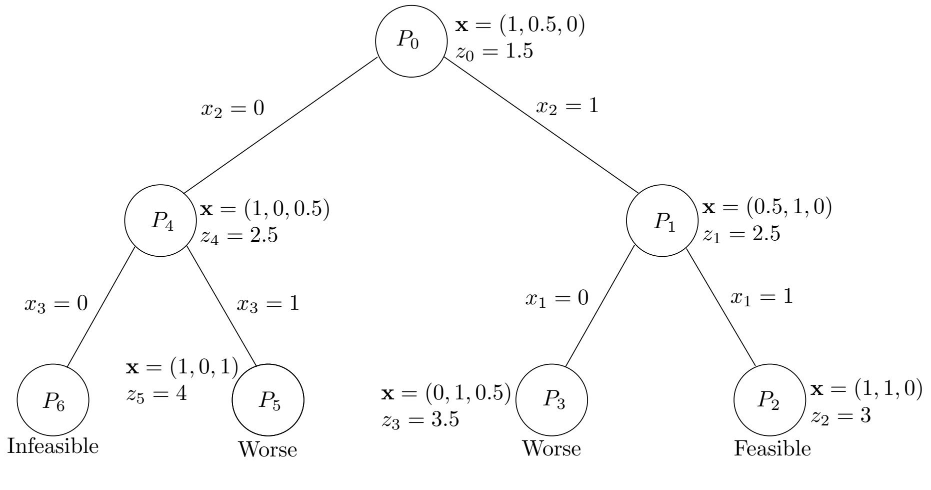

Figure 2: Search tree describing the branch-and-bound method. The original problem P_0 is divided into the sub problems P_i, $i = 1, \ldots, 6$. At every P_i, a relaxed problem is solved with one or more variables fixed, until a feasible solution is obtained. Then the remaining branches are examined and finally the optimal solution is found.

Even though this method guarantees that the obtained solution is optimal it still requires too much computing time for large problems. The number of sub problems grows exponentially with the number of variables, therefore sometimes the search is stopped when the difference between the upper and lower bound is small enough. Another way to reduce the number of subproblems is to use cutting planes.

Cutting-plane methods

Cutting-plane methods are another group of methods that are based on repeatedly solving LP relaxations of the IP problem. To understand the basic principle of a cutting-plane method the following definitions are needed.

Definition 8. *A point* $\mathbf{y}$ *is a convex combination of the points* $\mathbf{x}^k$*,* $k \in K$*, if* $\mathbf{y} = \sum_{k\in K} \lambda_k \mathbf{x}^k$*,* $\sum_{k\in K} \lambda_k = 1$*, and* $\lambda_k \geq 0$*,* $\forall\, k \in K$*.*

Definition 9. *The convex hull of a set of points* $\mathbf{x}^k$*,* $k \in K$*, consists of all possible convex combinations of these points.*

The convex hull of an arbitrary subset of $\mathbb{R}^n$ is the smallest convex set that includes this subset. To get an intuitive understanding, consider the following set.

$$\left\{ \begin{array}{rcl} 5x_1 + 4x_2 & \leq & 32 \\ -17x_1 + 9x_2 & \leq & 9 \\ x_1, x_2 & \geq & 0\,, \quad \text{integer} \end{array} \right. \tag{8}$$

Figure 3 illustrates the convex hull of this set. Recall that for an IP problem the feasible set is defined by linear inequalities and equalities, and the requirement for variables to be integers. All points in such a set will be contained in the convex hull, but no new feasible points will be added.

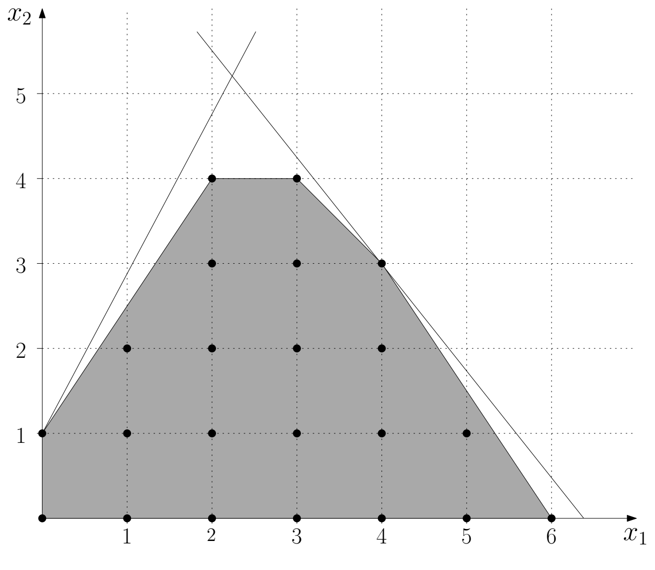

Figure 3: The figure illustrates the set (8) with its convex hull shaded. The two lines represent the inequalities and the dots represent the discrete points in the set.

Definition 10. *Let X denote the feasible set of some IP problem. A valid inequality is a linear inequality, $\sum_{j=1}^{n} a_j x_j \leq b$, that is satisfied for every $\mathbf{x} \in X$.*

Since a valid inequality is satisfied for every feasible point, adding it as a constraint to the IP problem will not remove any feasible IP solution.

Now consider an IP problem with feasible region X. From the definitions we can make two important observations:

- Adding valid inequalities of the problem to X will approximate the convex hull of X, denoted by X_c.
- Solving the LP relaxation of the IP problem with $x \in X_c$ will produce the same solution as solving the IP problem with $x \in X$.

A cutting-plane method solves LP relaxations of an IP problem and in each new iteration it adds one or several valid inequalities, in this context called cuts. This will move the solution of the LP relaxation closer to the optimal IP solution. It is not necessary to make cuts until the entire convex hull has been found. For a cut to be meaningful it has to be made so that the current LP optimum is removed. Otherwise, the optimal LP solution will not change in the next iteration. There are many strategies for generating these cuts, see for example [6], [12] and [13].

Cutting planes can be used together with branch-and-bound to cut away some parts of the LP relaxations' feasible sets and reduce the computation load.

2.5 AMPL and CPLEX

AMPL ("A Mathematical Programming Language") is a computer language used for modeling and solving various types of large scale optimization problems. AMPL is well suited for these problems because the syntax closely resembles the mathematical notation used in optimization. For example, it is very simple to define a variable to have indices from different sets and then sum over one or several of these indices. Variables can be chosen to be continuous, integer or binary depending on the problem at hand.

To model an optimization problem in AMPL, two files are used: one model file and one data file. The model file contains definitions of variables, all constraints and the objective function, while the data file contains definitions of sets and values of the parameters that are used. To solve a problem, these files must be loaded into AMPL. Executing the solve command will then generate a specific optimization problem and, unless an error message stops the process, AMPL enters the presolve phase. In this stage AMPL attempts to simplify the problem, for example by finding constraints that fix a variable or eliminating constraints that are redundant, see [14] for more details. When this is done, the simplified problem is passed to one of many possible solvers.

The default solver for linear integer programs, and the one used in this report, is CPLEX. For this kind of problem CPLEX uses branch-and-cut [15], which is a combination of a branch-and-bound method and a cutting-plane method, which are both explained in Section 2.4. A branch-and-cut method iteratively solves LP relaxations of the IP problem and breaks it down as in a branch-and-bound method and adds valid inequalities so as to more closely approximate the IP problem [16]. There is a multitude of options available for CPLEX that allow the user to give instructions on how the solver should operate but since the problem solved in this report is relatively small and quickly solved by a modern computer it will not be necessary to adjust the default settings.

When the solver is finished it will return the optimal solution to AMPL which can then display all relevant information about the solution including objective value, variable values, which inequality constraints are active (i.e., holds with equality) etc.

3 Problem description

To solve the given timetabling problem, a model has to be constructed. All variables will be binary since they will essentially model yes-or-no decisions, i.e, if a course should be scheduled at a certain time on a certain day or not. The model then takes the form of an integer programming problem and may thus be solved as such in AMPL. First, some specifics of the model are discussed individually, such as students, teachers, courses, and rooms. After that, the description of all the used constraints are listed.

3.1 Students

There are different groups of students taking different courses, depending on what program and year they study. Instead of sets of different groups of students the model has subsets of the set of courses, where one subset contains all courses for one group of students.

3.2 Teachers

Each course is given by a certain teacher, assumed to be available at any time of the day. Teachers are basically connected to courses and therefore there are no variables for the teachers. In case a teacher has certain preferences, this is implemented in the objective function. By adding a weight on a certain time of a certain day for the course which is taught by the teacher, the course will not be scheduled at this time if possible.

3.3 Courses

There is a certain number of courses to be scheduled, depending on the study period. Every course has a certain number of lectures, exercises, and computer labs during that study period. Each course is identified by a course code and has a certain number of students who can and will attend it. All the courses are split up into different course groups containing courses taken by the same group of students. These groups are the courses for the students studying the first year of the mathematics program at the University of Gothenburg, the second year of the mathematics program at the University of Gothenburg, the first year of Engineering mathematics and computational science at Chalmers University of Technology and the second year of Engineering mathematics and computational science at Chalmers University of Technology. Collisions between courses are not allowed within any of these groups.

There are two additional groups, *advanced courses* and *other courses. Advanced courses* are courses for students studying the third year of the mathematics program at the University of Gothenburg. The lectures in these courses are not allowed to collide but exercises and computer labs are. This is due to the high number of courses in this group. *Other courses* includes various remaining courses that are allowed to collide with any course except themselves. For a few courses the exercises are split into smaller groups. Also, some courses are scheduled together with other departments, in the rooms of the mathematics department. These are fixed and may not be changed. In the actual timetable, seen in Figure 1, there are some courses with sessions scheduled in rooms not belonging to the mathematics department. In the model, these sessions will be scheduled in the rooms that do belong to the mathematics department.

3.4 Rooms

There is a certain number of rooms available with different capacities for students. The rooms and their capacities stay the same for every study period. There are ten rooms for lectures and eight rooms for exercises. The exercise rooms are used as lecture rooms as well and are thus included in the ten lecture rooms. There are some special computer rooms which are used for computer labs. In the mathematical formulation, the simplification is made that there is only one large computer room. This was done because it is a fairly common occurrence that all computer rooms are booked for one lab session. Only a few rooms are already occupied by courses that cannot be changed or moved. Otherwise, the rooms are free and can be used at any time.

3.5 Constraints

There are two groups of constraints to be taken into consideration, hard constraints and soft constraints, as described in Section 2.1. The constraints were formulated based on information obtained from those responsible for the scheduling at the mathematics department. The hard constraints are necessary to get a valid timetable and the soft constraints reflect the preferences of the majority.

Hard constraints

1. There must be no more than one lecture, exercise or computer lab in one room at one time.
2. The timetable has to be complete, i.e., all courses must be scheduled with the required number of sessions each week.
3. The rooms must be large enough for the courses.
4. No course is allowed to collide with itself.
5. Collisions between courses in the same group are not allowed, except for *advanced courses* and *other courses*.
6. Collisions between lectures in *advanced courses* are not allowed.
7. Lectures, exercises and computer labs must be given in their respective corresponding rooms.

Soft constraints

1. There should be no more then one lecture, one exercise, and one computer lab per day and course.
2. The timetable for the different groups of students should be spread over the week and as compact as possible during each day.
3. An exercise in a course should be given immediately after a lecture in the same course.
4. Every lecture in a course should be in the same room. This applies to exercises as well.
5. Monday morning and Friday afternoon should not contain any sessions.
6. Sessions should preferably be scheduled between 10:00 to 15:00.

4 Mathematical formulation

This section contains the mathematical description of the problem and a list of all included sets, parameters, variables and constraints. At the mathematics department of the University of Gothenburg and Chalmers University of Technology, every year is divided into four study periods, where each period spans eight weeks. The rooms are the same in each period but the courses are different. The teaching times are from Monday to Friday, 8:00 to 17:00, where each day is divided into four time periods. Between each time period there is at least a 15 minute break.

The model is made in such a way that it is possible to make a timetable for any study period by replacing the courses and adding the new ones to the correct sets.

4.1 Model

Listed below are all sets, parameters, variables, and constraints used in the optimization model. The AMPL model file used for the first study period is presented in Appendix B and the MATLAB code used to visualize the solution is presented in Appendix C.

Sets

- $\mathcal{D} = \{1, 2, 3, 4, 5\}$ The days from Monday until Friday.
- $\mathcal{P} = \{1, 2, 3, 4\}$ The time periods of a day where 1 is 8:00-9:45, 2 is 10:00-11:45, 3 is 13:15-15:00 and 4 is 15:15-17:00.
- $\mathcal{C}_{GU1}$ ={course#1, ..., course#$|\mathcal{C}_{GU1}|$} Courses given to first year students of the mathematics program at the University of Gothenburg.
- $\mathcal{C}_{GU2}$ ={course#1, ..., course#$|\mathcal{C}_{GU2}|$} Courses given to second year students of the mathematics program at the University of Gothenburg.
- $\mathcal{C}_{EM1}$ ={course#1, ..., course#$|\mathcal{C}_{EM1}|$} Courses given to first year students of Engineering mathematics and computational science at Chalmers University of Technology.
- $\mathcal{C}_{EM2}$ ={course#1, ..., course#$|\mathcal{C}_{EM2}|$} Courses given to second year students of Engineering mathematics and computational science at Chalmers University of Technology.
- $\mathcal{C}_{adv}$ ={course#1, ..., course#$|\mathcal{C}_{adv}|$} Advanced courses.
- $\mathcal{C}_{others}$ ={course#1, ..., course#$|\mathcal{C}_{others}|$} Other courses.
- $\mathcal{C} = \mathcal{C}_{GU1} \cup \mathcal{C}_{GU2} \cup \mathcal{C}_{EM1} \cup \mathcal{C}_{adv} \cup \mathcal{C}_{others}$ All courses.
- $\mathcal{C}_g$ The set of courses split into two or more exercise groups, $\mathcal{C}_g \subseteq \mathcal{C}$.
- $\mathcal{C}_1$ The set of courses with more than 3 lectures per week, $\mathcal{C}_1 \subseteq \mathcal{C}$.
- $\mathcal{C}_2$ The set of courses with more exercises than lectures per week $\mathcal{C}_2 \subseteq \mathcal{C}$.
- $\mathcal{C}_3$ The set of courses with more than five exercises per week $\mathcal{C}_3 \subseteq \mathcal{C}$.
- $\mathcal{C}_4$ The set of courses with more than one computer lab per day, $\mathcal{C}_4 \subseteq \mathcal{C}$.
- $\mathcal{C}_5$ The set of courses with more than five exercises per week, without being fixed, $\mathcal{C}_5 \subseteq \mathcal{C}$.

- $\mathcal{R}_{lec}$ ={room#1, ..., room#$|\mathcal{R}_{lec}|$} The rooms used for lectures.
- $\mathcal{R}_{ex}$ ={room#1, ..., room#$|\mathcal{R}_{ex}|$} The rooms used for exercises.
- $\mathcal{R}_{com}$ ={room#1, ..., room#$|\mathcal{R}_{com}|$} The rooms with computers used for computer labs.
- $\mathcal{R} = \mathcal{R}_{lec} \cup \mathcal{R}_{ex} \cup \mathcal{R}_{com}$ All rooms.

Parameters

- s_c The course size, i.e., the number of students, for course $c \in \mathcal{C}$.
- m_r The room capacity for room $r \in \mathcal{R}$.
- $n_{lec,c}$ The number of lectures for course $c \in \mathcal{C}$.
- $n_{ex,c}$ The number of exercises for course $c \in \mathcal{C}$.
- $n_{com,c}$ The number of computer labs for course $c \in \mathcal{C}$.
- g_c The number of exercise groups for course $c \in \mathcal{C}$.

Variables

- $x_{d,p,c,r}$ Binary variable where $d \in \mathcal{D}, p \in \mathcal{P}, c \in \mathcal{C}$ and $r \in \mathcal{R}$. The variable takes the value 1 if course c has a lecture on day d, at time p, in room r. Otherwise it is 0.
- $y_{d,p,c,r}$ Binary variable where $d \in \mathcal{D}, p \in \mathcal{P}, c \in \mathcal{C}$ and $r \in \mathcal{R}$. The variable takes the value 1 if course c has an exercise on day d, at time p, in room r. Otherwise it is 0.
- $z_{d,p,c,r}$ Binary variable where $d \in \mathcal{D}, p \in \mathcal{P}, c \in \mathcal{C}$ and $r \in \mathcal{R}$. The variable takes the value 1 if course c has a computer lab on day d, at time p, in room r. Otherwise it is 0.
- $w1_{c,r}$ Binary variable where $c \in \mathcal{C}$ and $r \in \mathcal{R}$. The variable takes the value 1 if course c has a lecture in room r. Otherwise it is 0. This variable is used to force the lectures of course c to always be in the same room.
- $w2_{c,r}$ Binary variable where $c \in \mathcal{C}$ and $r \in \mathcal{R}$. The variable takes the value 1 if course c has an exercise in room r. Otherwise it is 0. This variable is used to force the exercises of course c to always be in the same room.

Constraints

$$x_{d,p,c,r} \cdot s_c \leq m_r, \qquad d \in \mathcal{D},\, p \in \mathcal{P},\, c \in \mathcal{C},\, r \in \mathcal{R}_{lec}, \tag{9}$$

$$0.8 \cdot y_{d,p,c,r} \cdot s_c / g_c \leq m_r, \qquad d \in \mathcal{D},\, p \in \mathcal{P},\, c \in \mathcal{C},\, r \in \mathcal{R}_{ex}, \tag{10}$$

$$z_{d,p,c,r} \cdot s_c \leq m_r, \qquad d \in \mathcal{D},\, p \in \mathcal{P},\, c \in \mathcal{C},\, r \in \mathcal{R}_{com}, \tag{11}$$

$$\sum_{c \in \mathcal{C}} (x_{d,p,c,r} + y_{d,p,c,r} + z_{d,p,c,r}) \leq 1, \qquad d \in \mathcal{D},\, p \in \mathcal{P},\, r \in \mathcal{R}, \tag{12}$$

$$\sum_{c\in\mathcal{C}_{GU1}}\sum_{r\in\mathcal{R}}(x_{d,p,c,r}+y_{d,p,c,r}/g_c+z_{d,p,c,r})\leq 1, \qquad d\in\mathcal{D},\, p\in\mathcal{P}, \tag{13}$$

$$\sum_{c\in\mathcal{C}_{GU2}}\sum_{r\in\mathcal{R}}(x_{d,p,c,r}+y_{d,p,c,r}/g_c+z_{d,p,c,r})\leq 1, \qquad d\in\mathcal{D},\, p\in\mathcal{P}, \tag{14}$$

$$\sum_{c\in\mathcal{C}_{EM1}}\sum_{r\in\mathcal{R}}(x_{d,p,c,r}+y_{d,p,c,r}/g_c+z_{d,p,c,r})\leq 1, \qquad d\in\mathcal{D},\, p\in\mathcal{P}, \tag{15}$$

$$\sum_{c\in\mathcal{C}_{EM2}}\sum_{r\in\mathcal{R}}(x_{d,p,c,r}+y_{d,p,c,r}/g_c+z_{d,p,c,r})\leq 1, \qquad d\in\mathcal{D},\, p\in\mathcal{P}, \tag{16}$$

$$\sum_{c\in\mathcal{C}_{adv}}\sum_{r\in\mathcal{R}}(x_{d,p,c,r})\leq 1, \qquad d\in\mathcal{D},\, p\in\mathcal{P}, \tag{17}$$

$$\sum_{r\in\mathcal{R}}(x_{d,p,c,r}+y_{d,p,c,r}/g_c+z_{d,p,c,r})\leq 1, \qquad d\in\mathcal{D},\, p\in\mathcal{P}\ c\in\mathcal{C}_{others}, \tag{18}$$

$$\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}_{lec}} x_{d,p,c,r}=n_{lec,c}, \qquad c\in\mathcal{C}, \tag{19}$$

$$\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}_{ex}} y_{d,p,c,r}=n_{ex,c}\cdot g_c, \qquad c\in\mathcal{C}, \tag{20}$$

$$\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}_{com}} z_{d,p,c,r}=n_{com,c}, \qquad c\in\mathcal{C}, \tag{21}$$

$$\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}}(x_{d,p,c,r}+x_{d+1,p,c,r})\leq 1, \qquad d\in\mathcal{D}\backslash\{5\},\, c\in\mathcal{C}\backslash\{C_1\}, \tag{22}$$

$$\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}} x_{d,p,c,r}-w1_{c,r}\cdot n_{lec,c}=0, \qquad c\in\mathcal{C},\, r\in\mathcal{R}_{lec}, \tag{23}$$

$$\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}} x_{d,p,c,r}-w2_{c,r}\cdot n_{ex,c}=0, \qquad c\in\mathcal{C}\backslash\{C_g\},\, r\in\mathcal{R}_{ex}, \tag{24}$$

$$\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}(y_{d,p,c,r1}+y_{d,p,c,r2})-w2_{c,r1}\cdot n_{ex,c}-w2_{c,r2}\cdot n_{ex,c}=0, \tag{25}$$
$$c\in\mathcal{C}_g,\ r1\in\mathcal{R}_{ex},\ r2\in\mathcal{R}_{ex},$$

$$\sum_{r\in\mathcal{R}_{lec}}(y_{d,p,c,r}/g_c-x_{d,p-1,c,r})\leq 0, \qquad d\in\mathcal{D},\ p\in\mathcal{P}\backslash\{1\},\ c\in\mathcal{C}\backslash\{C_2\}, \tag{26}$$

$$\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}}x_{d,p,c,r}\leq 1, \qquad d\in\mathcal{D},\ c\in\mathcal{C}, \tag{27}$$

$$\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}}y_{d,p,c,r}\leq 1, \qquad d\in\mathcal{D},\ c\in\mathcal{C}\backslash\{C_g,C_3\}, \tag{28}$$

$$\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}}y_{d,p,c,r}\leq 2, \qquad d\in\mathcal{D},\ c\in\{C_5\}, \tag{29}$$

$$\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}}z_{d,p,c,r}\leq 1, \qquad d\in\mathcal{D},\ c\in\mathcal{C}\backslash\{C_4\}, \tag{30}$$

Explanation of the constraints

(9) The rooms must be at least as large as the course size for the lectures.

(10) The rooms must be large enough for the exercises. There is a multiplicative factor of 0.8 because normally not all the students take part in the exercises.

(11) The rooms must be at least as large as the course size for the computer labs.

(12) There must be no more than one lecture/exercise/computer lab in a given room at a given time.

(13) Courses in $\mathcal{C}_{GU1}$ must not collide.

(14) Courses in $\mathcal{C}_{GU2}$ must not collide.

(15) Courses in $\mathcal{C}_{EM1}$ must not collide.

(16) Courses in $\mathcal{C}_{EM2}$ must not collide.

(17) Lectures of courses in $\mathcal{C}_{adv}$ must not collide.

(18) Courses in $\mathcal{C}_{others}$ must not collide with themselves.

(19)-(21) Every course must have the required number of lectures, exercises, and computer labs.

(22) Lectures must be spread out over the week, except for courses in $\mathcal{C}_1$.

(23) For each course every lecture must be in the same room.

(24) For each course every exercise must in the same room.

(25) For each course where the exercises are split up into groups, each group's sessions must be in the same room.

(26) Exercises must be after a lecture in the same course or in time period 1.

(27) There must be no more than one lecture per day and course.

(28) There must be no more than one exercise per day and course, except for courses in C_3.

(29) There must be no more than two exercises per day for courses in C_5.

(30) There must be no more than one computer lab per day and course, except for courses in C_4.

Objective function
The objective function is used to set preferences for the different periods of the day. Since the constraints will always ensure that the resulting timetable is valid, the objective function can be chosen to reflect whatever preferences the majority has. This is done by assigning weights to the variables corresponding to the different time periods. As this is a minimization problem, a large weight will increase the objective function and will therefore not be beneficial to the solution. Thus the weights control the result and the way they are chosen is essential and will be discussed in Section 6. The objective funtion is specified as follows

$$\begin{aligned}\min f = &\sum_{d\in\mathcal{D}}\sum_{c\in\mathcal{C}}\sum_{r\in\mathcal{R}}(x_{d,1,c,r} + x_{d,3,c,r} + 4x_{d,4,c,r} + 3y_{d,1,c,r} + y_{d,2,c,r} \\ &\qquad + 2y_{d,4,c,r} + 3z_{d,1,c,r} + z_{d,2,c,r} + 2z_{d,4,c,r}) \\ &+\sum_{c\in\mathcal{C}}\sum_{r\in\mathcal{R}} 5(x_{1,1,c,r} + y_{1,1,c,r} + z_{1,1,c,r} + x_{5,4,c,r} + y_{5,4,c,r} + z_{5,4,c,r}).\end{aligned} \tag{31}$$

The triple sum is over courses, days and rooms, and controls lectures, exercises and computer labs. Lectures are preferably held in time period 2, which is why $x_{d,2,c,r}$ is not included in the sum, i.e., it has a weight of zero. If this is not possible they should be in time period 1 or 3, which is why these periods have a weight of one. The least preferred time period for lectures is time period 4, which has a weight of four, thus it will only be chosen if no other better solution is possible. Exercises and computer labs are preferably held in time period 3 and then in descending order in time period 2, 4, and 1.

The double sum assigns large weights to Monday time period 1 and Friday time period 4 because these are the two least preferred times.

4.2 Data

In this section only the data for the first study period of spring 2013 is presented, data for the second period is included in Appendix A. Table 1 includes all data about the rooms, that is, names, sizes and types. This data can be used for any study period, as the rooms, their sizes and their types do not change.

Name	Type	Size
Euler	lecture	70
Pascal	lecture	60
MVF21	lecture/exercise	30
MVF23	lecture/exercise	30
MVF26	lecture/exercise	36
MVF31	lecture/exercise	42
MVF32	lecture/exercise	16
MVF33	lecture/exercise	36
MVH11	lecture/exercise	26
MVH12	lecture/exercise	38
MVF22	computer lab	90

Table 1: Rooms available at the department of Mathematical Sciences

Table 2 includes all data for the courses, that is, their sizes and their number of lectures, exercises, and computer labs, how many exercise groups each course is split up to, in which sets the courses are included, and whether they are fixed. In total, there are 78 sessions. This data is different for each study period according to the offered courses. The course sizes used are the actual number of registered students for the courses during the spring of 2013.

Course code	Course groups	Size	Lectures	Exercises	Computer labs	No. of groups	Fixed
MMG720	$\mathcal{C}_{adv}$	8	2	1	0	1	No
MMG300	$\mathcal{C}_{GU1}$	38	2	2	0	1	Yes
MVG300	$\mathcal{C}_{GU1}$	34	3	0	3	1	Yes
MMG800	$\mathcal{C}_{EM1}$, $\mathcal{C}_{adv}$	55	4	0	0	1	No
MSG200	$\mathcal{C}_{GU2}$, $\mathcal{C}_{EM1}$	50	2	2	0	1	No
MMA511	$\mathcal{C}_{EM1}$, $\mathcal{C}_{adv}$	13	3	0	0	1	No
MMG500	$\mathcal{C}_{GU2}$, $\mathcal{C}_{EM2}$, $\mathcal{C}_{adv}$	28	2	2	0	2	No
MSA200	$\mathcal{C}_{EM2}$	11	2	1	0	1	No
MMA421	$\mathcal{C}_{EM2}$, $\mathcal{C}_{adv}$	19	2	1	0	1	No
MSG830	$\mathcal{C}_{others}$	44	2	1	2	1	Yes
MMGF20	$\mathcal{C}_{others}$	29	2	2	0	1	No
MMGK11	$\mathcal{C}_{others}$	49	4	4	0	2	No
MMGL31	$\mathcal{C}_{others}$	20	4	6	2	1	No
LGMA10	$\mathcal{C}_{others}$	35	4	8	0	1	Yes
MMGF30	$\mathcal{C}_{others}$	21	3	0	0	1	Yes

Table 2: Courses in the first study period of spring 2013

5 Results

Using the data in Table 1 and 2 in the model and visualizing the results gives the timetable in Figure 4. The figure shows a timetable for one arbitrary week, consisting of five days with four time periods per day. The timetable in the figure contains all courses for which a timetable is sought, as well as the rooms the courses are scheduled in. To clarify the timetable, and to make it easier to examine whether the constraints are fulfilled within each group, timetables consisting of smaller groups of courses are given in Figure 5 - 8.

In Figure 5 all courses given to first and second year students of the mathematics program at the University of Gothenburg are shown. The courses are given a color to distinguish first year students from second year students. Courses colored with color 1 are the courses given to first year students and courses colored with color 2 are the courses given to second year students. The same is done for students of Engineering mathematics and computational science at Chalmers University of Technology in Figure 6. Here color 3 is for first year students and color 4 is for second year students. In Figure 7 and 8 *advanced courses* and *other courses* are shown respectively.

From Figure 5 and Figure 6 it is easy to see that no courses with the same color collide, i.e., courses within the same group do not collide. In Figure 7 we see that no lectures collide but that some exercises collide with lectures of other courses, which is allowed in the *advanced courses* group according to the model. For *other courses* collisions are allowed, as seen in Figure 8. All figures show that no course collides with itself. Hence hard constraints 4, 5 and 6 are fulfilled.

Comparing the timetables with the data in Table 1 and Table 2, we find that all courses are scheduled with their required number of sessions (hard constraint 2). We also notice that all rooms are large enough for the corresponding course size and that all sessions are scheduled in the right type of room (hard constraints 3 and 7). There is at most one course in one room at one time (hard constraint 1). Hence all hard constraints are fulfilled.

Further inspection of the timetables shows that lectures within most courses are spread out over the week, i.e., these courses do not have lectures on two consecutive days (soft constraint 2). This does not hold for courses with more than three lectures per week, i.e., MMG800, MMGK11, MMGL31, and LGMA10, all of which have four lectures per week. We also see in Figure 4 that exercises always follow directly after a corresponding lecture (soft constraint 3), except for the courses MMGL31 and LGMA10. This is because these courses have more exercises than lectures and because LGMA10 is fixed.

Figure 4 also shows that there is at most one lecture per day and course. For some courses this is true also for exercises and computer labs (soft constraint 1). However, some courses have more than five exercises each week and some are fixed. We see that all lectures in a course are scheduled in the same room throughout the week. This is fulfilled for exercises and computer labs as well (soft constraint 4). The only sessions given in the first time period on Monday morning and last time period on Friday are fixed (soft constraint 5). Finally, as many sessions as possible are scheduled in time period 2 and 3 (soft constraint 6). All soft constraints have thereby been met, as far as it is possible.

Data and results for the second study period of spring 2013 are included in Appendix A and can be validated in the same fashion.

	Monday		Tuesday		Wednesday		Thursday		Friday	
8:00	LGMA10 ex	MVF31	MMG800 lec	Pascal	MMA511 lec	MVF33	MMG800 lec	Pascal	MMGL31 lec	MVF26
			LGMA10 ex	MVF31			LGMA10 ex	MVF31	MMA511 lec	MVF33
			MMGL31 ex	MVF33			MMGL31 ex	MVF33	LGMA10 ex	MVF31
10:00	MMGK11 lec	Euler	MSG830 lec	Euler	MMGK11 lec	Euler	MMGK11 lec	Euler	MMGK11 lec	Euler
	MSA200 lec	MVF21	MMGF30 lec	MVF23	MMGF20 lec	MVF23	MMGL31 lec	MVF26	MSA200 lec	MVF21
	MMGF20 lec	MVF23	MMGL31 lec	MVF26	MMGL31 lec	MVF26	MMG500 lec	MVH12	MMGF30 lec	MVF23
	MMG800 lec	Pascal	MMG500 lec	MVH12	MMA421 lec	MVH12	MMG300 lec	Pascal	MMG720 lec	MVF26
	LGMA10 ex	MVF31	MMG300 lec	Pascal	MSG200 lec	Pascal	LGMA10 ex	MVF31	MSG200 lec	Pascal
			LGMA10 ex	MVF31	MSG830 ex	MVF33			LGMA10 ex	MVF31
									MMGL31 ex	MVF33
13:15	MVG300 lec	Euler	MMG720 lec	MVF26	MVG300 lec	Euler	MSG830 lec	Euler	MVG300 lec	Euler
	MMA511 lec	MVF33	LGMA10 lec	Pascal	MMGF30 lec	MVF23	LGMA10 lec	Pascal	MMA421 lec	MVH12
	LGMA10 lec	Pascal	MMG500 ex	MVF23	MMGK11 ex	MVF21	MMGK11 ex	MVF21	LGMA10 lec	Pascal
	MMGK11 ex	MVF21	MMG500 ex	MVF31	MMA421 ex	MVF26	MMG500 ex	MVF23	MMGK11 ex	MVF21
	MSA200 ex	MVF26	MMGL31 ex	MVF33	MSG200 ex	MVF31	MMG500 ex	MVF31	MMG720 ex	MVF23
	MMGK11 ex	MVH11	MMG300 ex	MVH12	MMGL31 ex	MVF33	MMGL31 ex	MVF33	MSG200 ex	MVF31
	MMGF20 ex	MVH12	MSG830 com	MVF22	MMGK11 ex	MVH11	MMGK11 ex	MVH11	MMGK11 ex	MVH11
	MMGL31 com	MVF22			MMGF20 ex	MVH12	MMG300 ex	MVH12	MMGL31 com	MVF22
15:15	MVG300 com	MVF22	MSG830 com	MVF22	MMG800 lec	Pascal			MVG300 com	MVF22
					MVG300 com	MVF22				

Figure 4: Resulting timetable for all courses of the first study period of spring 2013.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	Color 1: 🟩 Color 2: 🟦				
10:00		MMG500 lec MVH12 MMG300 lec Pascal	MSG200 lec Pascal	MMG500 lec MVH12 MMG300 lec Pascal	MSG200 lec Pascal
13:15	MVG300 lec Euler	MMG500 ex MVF23 MMG500 ex MVF31 MMG300 ex MVH12	MVG300 lec Euler MSG200 ex MVF31	MMG500 ex MVF23 MMG500 ex MVF31 MMG300 ex MVH12	MVG300 lec Euler MSG200 ex MVF31
15:15	MVG300 com MVF22		MVG300 com MVF22		MVG300 com MVF22

Figure 5: The timetable for courses given to students at the mathematics program at the University of Gothenburg. Color 1 is the courses given to first year students and color 2 is the courses given to second year students.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	Color 3: 🟥 Color 4: 🟨	MMG800 lec Pascal	MMA511 lec MVF33	MMG800 lec Pascal	MMA511 lec MVF33
10:00	MSA200 lec MVF21 MMG800 lec Pascal	MMG500 lec MVH12	MMA421 lec MVH12 MSG200 lec Pascal	MMG500 lec MVH12	MSA200 lec MVF21 MSG200 lec Pascal
13:15	MMA511 lec MVF33 MSA200 ex MVF26	MMG500 ex MVF23 MMG500 ex MVF31	MMA421 ex MVF26 MSG200 ex MVF31	MMG500 ex MVF23 MMG500 ex MVF31	MMA421 lec MVH12 MSG200 ex MVF31
15:15			MMG800 lec Pascal		

Figure 6: The timetable for courses given to students of Engineering mathematics and computational science at Chalmers University of Technology. Color 3 is the courses given to first year students and color 4 is the courses given to second year students.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:00		MMG800 lec Pascal	MMA511 lec MVF33	MMG800 lec Pascal	MMA511 lec MVF33
10:00	MMG800 lec Pascal	MMG500 lec MVH12	MMA421 lec MVH12	MMG500 lec MVH12	MMG720 lec MVF26
13:15	MMA511 lec MVF33	MMG720 lec MVF26 MMG500 ex MVF23 MMG500 ex MVF31	MMA421 ex MVF26	MMG500 ex MVF23 MMG500 ex MVF31	MMA421 lec MVH12 MMG720 ex MVF23
15:15			MMG800 lec Pascal		

Figure 7: Courses in the group *advanced courses*.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	LGMA10 ex MVF31	LGMA10 ex MVF31 MMGL31 ex MVF33		LGMA10 ex MVF31 MMGL31 ex MVF33	MMGL31 lec MVF26 LGMA10 ex MVF31
10:00	MMGK11 lec Euler MMGF20 lec MVF23 LGMA10 ex MVF31	MSG830 lec Euler MMGF30 lec MVF23 MMGL31 lec MVF26 LGMA10 ex MVF31	MMGK11 lec Euler MMGF20 lec MVF23 MMGL31 lec MVF26 MSG830 ex MVF33	MMGK11 lec Euler MMGL31 lec MVF26 LGMA10 ex MVF31	MMGK11 lec Euler MMGF30 lec MVF23 LGMA10 ex MVF31 MMGL31 ex MVF33
13:15	LGMA10 lec Pascal MMGK11 ex MVF21 MMGK11 ex MVH11 MMGF20 ex MVH12 MMGL31 com MVF22	LGMA10 lec Pascal MMGL31 ex MVF33 MSG830 com MVF22	MMGF30 lec MVF23 MMGK11 ex MVF21 MMGL31 ex MVF33 MMGK11 ex MVH11 MMGF20 ex MVH12	MSG830 lec Euler LGMA10 lec Pascal MMGK11 ex MVF21 MMGL31 ex MVF33 MMGK11 ex MVH11	LGMA10 lec Pascal MMGK11 ex MVF21 MMGK11 ex MVH11 MMGL31 com MVF22
15:15		MSG830 com MVF22			

Figure 8: Courses in the group *other courses*.

6 Analysis and discussion

In order to examine the model, sensitivity analysis was performed. In addition to this, the general features of the resulting timetable were studied and discussed with respect to constraints and the current timetable.

6.1 Sensitivity analysis

The optimal solution to a problem is only optimal with respect to the given data. Sensitivity analysis is the study of how specific constraints and variables affect the solution. There are not a lot of concepts behind sensitivity analysis for scheduling problems. This made the analysis quite limited and the actual performed analysis merely involved observations of how the model responds to manual changes of the constraints and the objective function.

In order to examine what effect the different constraints have on the resulting timetable, some constraints where temporarily removed and the resulting timetable studied. It should be clarified that this was never done with the hard constraints, since a removed hard constraint would only yield a nonvalid result.

When removing the constraint saying that all lectures in a course should be in the same room (soft constraint 4) we found that this has minimal effect on the solution. This indicates that this constraint can be used without risk of loosing otherwise valid and favourable solutions. In this particular case, a possible explanation to this is the high number of rooms with respect to the number of courses.

If some rooms are removed the model still gives a solution, but the constraint saying that exercises should be scheduled immediately after a corresponding lecture (soft constraint 3) is broken. Also, the constraint saying that all lectures in a course should be scheduled in the same room (soft constraint 4) is now responsible for a longer computation time, even though the resulting timetable does not differ with or without this constraint.

The major restriction in the model is the number of courses that are not allowed to collide. If all courses were allowed to collide, it would be possible to schedule up to 220 sessions a week (4 time periods, 5 days, 11 rooms). Including the constraints that some courses cannot collide significantly decreases this number.

The objective function (31) has been chosen to match the preferences from teachers and students, by focusing lectures to the two intermediate time periods. By changing the objective function it is possible to make the timetable focused on any two time periods. For example, instead of weighting time period 4 with high values as is done now, it would be possible to weight time period 1 and 2 with high values, which would result in a timetable with most of its sessions in the afternoon. Hence it is possible to steer the timetable by using these weights, which increases the adaptabillity to newly added preferences.

It is also possible to remove the second term of the objective function, to allow more courses on Monday morning and Friday afternoon. In the same way, this can be changed to weight courses on any other day and time period, hence making the timetable look a bit different.

6.2 Discussion

One of the major challenges with scheduling, regardless of whether it is done with software or by hand, is determining what qualifies as a ”good” schedule. Some soft constraints will probably be unanimously agreed upon, for example that the lectures in a course should be spread out over the week. This gives students the opportunity to prepare for lectures in advance and to let the material sink in afterwards. However, in more subjective matters, such as at what times lectures should be scheduled, there are going to be different preferences and compromises will have to be made.

In this case study, according to the provided information, the general opinion among students and teachers was that having courses scheduled in time period 2 and 3 is preferred. Most of them also agreed that, when possible, exercises should be scheduled directly after lectures. This is what was considered when the model was formulated.

Comparing the obtained solution of the formulated model (visualized in the Figure 4) with the current schedule (Figure 1) reveals some similarities, most noticeably that most courses are concentrated to the two intermediate time periods (10:00-11:45 and 13:15-15:00). In fact, when the model is rewritten so that all courses are fixed in their actual positions, almost every constraint can be included. This indicates that the actual timetable is made with respect to much of the same restrictions as the integer programming model. Most of the constraints that are violated strictly have to do with preferences, e.g., that exercises should follow after lectures, that two lectures in a course should not be given on consecutive days and that lectures (and excercises) should be given in the same room. Some courses are also scheduled with two computer labs in one day but this is something that most students probably do not find inconvenient.

The most relevant difference between the suggested solution and the current timetable is that the current timetable does not take consideration to all courses in the *advanced courses* group, and consequently some of them do collide. In one sense this could be considered an improvement by the optimization approach. It does make it easier for students who take courses from this group to attend their lectures but might also bring some unwanted consequences. Since the constraints about which courses may not collide are the most restricting in this model, introducing even more conditions that courses should not collide will cause some sessions to be scheduled at times that are not desirable. For example, there might be combinations of courses in *advanced courses* that typically very few students take, e.g., students specializing in statistics are perhaps unlikely to take courses in computational science. Then it might be preferable to let two such courses collide. Which courses should be bound by these constraints is therefore a delicate matter. However, once it has been decided, it is of course easy to implement.

A difference that is a clear improvement is that sessions that are currently scheduled in rooms not belonging to the mathematics department have been moved to rooms that do belong to the mathematics department.

The timetabling problem solved in this report is not extremely large. Table 3 presents some basic statistics about the problems that were specified for the two study periods.

	Courses	Variables	Constraints	Non-zeros (constraints)	Computation time
First study period	15	10 230	7 012	59 772	1.02 s
Second study period	13	8 866	6 153	56 919	0.07 s

Table 3: Size of solved problems

These small scale problems can be solved to optimality by CPLEX without difficulty and with hardly any time requirement. The significant difference in computation time between the two study periods is partly explained by the number of courses and the number of sessions, which are both lower in the second study period. It is also noteworthy that the model can easily be adapted to specific data. It was used for both study periods without alterations.

7 Conclusions

The final timetable satisfies all hard constraints, listed in Section 3.5, which guarantees that it is in fact a valid timetable, and it could be applied as long as there are no restrictions from other departments. It is a good timetable in the sense that it fulfills the soft constraints to a higher degree than the current timetable. The model is general enough that it can be used for arbitrary data, and by changing the weights in the objective function it can be adapted to specific preferences. Solutions would have to be analyzed by the responsible personnel before they are used for real to be sure that all requirements are fulfilled but the work load would be reduced noticeably.

An IP model provides great flexibility for timetabling problems in that constraints can be formulated for restrictions that arise naturally for a university. For problems that are reasonably limited, branch-and-bound can guarantee optimality and the objective function can be adjusted until the resulting timetable is deemed satisfactory.

As of now, any changes to the model require basic knowledge of AMPL and MATLAB and advanced understanding of the model itself. A possible next step to facilitate the timetabling at the mathematics department would be to create a user friendly graphical interface. This would make it possible to make changes and immediately view the new timetable.

References

[1] Pinedo M. L. (2012) *Scheduling: Theory Algorithms and Systems.* Edition 4. Springer-Verlag New York Inc.

[2] Dantzig G. D. and Thapa M. N. (2003) *Linnear Programming: 2: Theory and Extensions.* Springer-Verlag New York Inc.

[3] Daskalaki S., Birbas T., and Housos E. (2004) *An integer programming formulation for a case study in university timetabling.* European Journal of Operational Research, vol. 153, issue 1, pp. 117-135.

[4] Dimopoulou M. and Miliotis P. (2001) *Implementation of a university course and examination timetabling system.* European Journal of Operational Research, vol 130, issue 1, pp. 202-213.

[5] Lewis R. (2007) *A survey of metaheuristic-based techniques for University Timetabling problems.* OR Spectrum, vol. 30, issue 1, pp. 167-190.

[6] Lundgren J., Rönnqvist M., and Värbrand P. (2010) *Optimization.* Lund: Studentlitteratur AB.

[7] Andréason N., Evgrafov A., and Patriksson M. (2005) *An Introduction to Continous Optimization.* Edition 1:2. Lund: Studentlitteratur AB.

[8] Thörnblad K., Strömberg A., Almgren T., and Patriksson M. (2010) *Optimization of schedules for a multitask production cell.* Proceedings of The 22nd NOFOMA Conference in Kolding, Denmark, 10-11th of June 2010.

[9] Przybylskia A., Gandibleuxa X., and Ehrgottb M. (2010) *A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives.* Discrete Optimization, vol 7, issue 3, pp. 149-165.

[10] Applegate D., Bixby R., Chvatal V., and Cook W. (2006) *The Traveling Salesman Problem: A Computational Study.* Princeton University Press.

[11] Hasle G. (2009) *Discrete Optimization - Heuristics.*
`http://www.sintef.no/project/eVITAmeeting/`
`Hasle%20Discrete%20Optimization%20-%20Heuristics.pdf` (2013-05-10)

[12] Cornuéjols G. (2008) *Valid inequalities for mixed integer linear programs.* Mathematical Programming, issue 112, pp. 3-44.

[13] Balas E., Ceria S., and Cornuéjols G. (1993) *A lift-and-project cutting plane algorithm for mixed 0-1 programs.* Mathematical Programming, issue 58, pp. 295-324.

[14] Fourer R., Gay, D. M., and Kernighan, B. W. (1993) *AMPL: A Modeling Language for Mathematical Programming.* Danvers, MA: Boyd & Fraser Pub. Co.

[15] *AMPL: A Modeling Language for Mathematical Programming* (2010) IBM ILOG AMPL Version 12.2 User's Guide.
`http://www.ampl.com/BOOKLETS/amplcplex122userguide.pdf` (2013-04-14)

[16] Mitchell J. E. (2002) *Branch-and-cut Algorithms for Combinatorial Optimization Problems.* In Handbook of Applied Optimization, P. Pardalos and M. G. C. Resende, editors, pp. 65-77. New York: Oxford University Press.

A Timetable for the second study period of spring 2013

The data for the second study period of spring 2013 is presented in Table 4 and the resulting timetable for all courses is shown in Figure 9.

Course code	Course groups	Size	Lectures	Exercises	Computer labs	No. of groups	Fixed
LGMA20	$\mathcal{C}_{others}$	30	2	4	1	1	Yes
MMGL32	$\mathcal{C}_{others}$	6	3	2	1	2	No
MMG511	$\mathcal{C}_{GU2}$, $\mathcal{C}_{EM1}$	65	3	0	0	1	No
MMG410	$\mathcal{C}_{GU1}$	46	3	0	3	1	Yes
MMG631	$\mathcal{C}_{GU2}$, $\mathcal{C}_{EM1}$	68	2	1	2	2	No
MMA130	$\mathcal{C}_{adv}$	4	3	0	0	1	No
MSA410	$\mathcal{C}_{EM1}$, $\mathcal{C}_{adv}$	25	2	0	0	1	No
MMA620	$\mathcal{C}_{EM2}$, $\mathcal{C}_{adv}$	36	2	0	2	1	Yes
MMA700	$\mathcal{C}_{EM1}$, $\mathcal{C}_{adv}$	51	2	1	0	2	No
MMA430	$\mathcal{C}_{EM2}$, $\mathcal{C}_{adv}$	25	2	1	0	1	No
MSF200	$\mathcal{C}_{EM2}$, $\mathcal{C}_{adv}$	11	4	0	0	1	No
MSA300	$\mathcal{C}_{EM1}$, $\mathcal{C}_{adv}$	18	2	0	2	1	No
MMG300	$\mathcal{C}_{GU1}$,	38	2	2	0	1	Yes

Table 4: Courses in the second study period of spring 2013.

In this study period there are 13 courses, two fewer than in the first study period. The total number of sessions is 54, which is lower than in the first study period. By studying Figure 9 it can be concluded that the same reasoning made for Figure 4 in Section 5 can be made for this timetable as well. In other words, all constraints are fulfilled and as many sessions as possible are scheduled in time period 2 and 3.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:00		MSA410 lec MVH11 LGMA20 ex MVF26	MMG511 lec Euler MSF200 lec MVF21	MSA300 lec MVF23	MMG511 lec Euler MMA130 lec Pascal LGMA20 ex MVF26
10:00	MMG511 lec Euler MMA430 lec MVF31 MMGL32 lec MVH12	MMG631 lec Euler MSF200 lec MVF21 MMG300 lec Pascal LGMA20 ex MVF26	MMGL32 lec MVH12 MMA700 lec Pascal LGMA20 com MVF22	MMG631 lec Euler MSF200 lec MVF21 MMG300 lec Pascal	MSF200 lec MVF21 MMGL32 lec MVH12 LGMA20 ex MVF26 MSA300 com MVF22
13:15	MMG410 lec Euler MMA130 lec Pascal MMA430 ex MVH11 MSA300 com MVF22	LGMA20 lec MVF26 MMA620 lec MVF33 MMG300 ex MVH12 MMG631 com MVF22	MMG410 lec Euler MMA130 lec Pascal MMA700 ex MVF26 MMGL32 ex MVF32 MMGL32 ex MVF33 MMA700 ex MVH11	MMA620 lec MVF33 MMG631 ex MVF21 MMG631 ex MVF31 MMG300 ex MVH12 MMGL32 com MVF22	MMG410 lec Euler LGMA20 lec MVF26 MMA430 lec MVF31 MMGL32 ex MVF32 MMGL32 ex MVF33 MMG631 com MVF22
15:15	MMA700 lec Pascal MMG410 com MVF22	MSA300 lec MVF23 MMA620 com MVF22	MMG410 com MVF22	MSA410 lec MVH11 MMA620 com MVF22	MMG410 com MVF22

Figure 9: Resulting timetable for all courses in the second study period of spring 2013.

B Sample of AMPL source code

The mathematical model had to be expressed in AMPL code. This is the model file used for the first study period of spring 2013.

```
set D; # Days
set P; # Periods
set C_GU1; # Courses taken by GU students first year
set C_GU2; # Courses taken by GU students second year
set C_EM1; # Courses taken by EM students first year
set C_EM2; # Courses taken by EM students second year
set C_adv; # Other courses that should not collide
set C_others; # All other courses
set C_g; # Set with courses that have their exercises splitted into small groups
set C := C_GU1 union C_GU2 union C_EM1 union C_EM2 union C_adv union C_others;
# All courses
set R_ex; # Exercise rooms
set R_lec; # Lecture rooms
set R_com; # Computer rooms
set R := R_ex union R_lec union R_com; # All rooms

param s{C}; # Course size
param m{R}; # Room capacity
param n_com{C}; # Number of computer labs
param n_lec{C}; # Number of lectures
param n_ex{C}; # Number of excercises
param g{C}; # Number of groups for exercises

var x{D,P,C,R} binary; # Lectures
var y{D,P,C,R} binary; # Excercises
var z{D,P,C,R} binary; # Computer labs

var w1{C,R} binary; # Help variable to force lectures to be in the same room
var w2{C,R} binary; # Help variable to force excercises to be in the same room

# Object function
minimize f: sum{d in D, c in C, r in R} (x[d,1,c,r] + x[d,3,c,r] + 4*x[d,4,c,r]
            + 3*y[d,1,c,r] + y[d,2,c,r] + 2*y[d,4,c,r] + 3*z[d,1,c,r]
            + z[d,2,c,r] + 2*z[d,4,c,r])
            + sum{c in C, r in R} 5*(x[1,1,c,r] + y[1,1,c,r] + z[1,1,c,r]
            + x[5,4,c,r] + y[5,4,c,r] + z[5,4,c,r]);

subject to
# Make sure that the classes fits in the rooms
Room_capacity_lec{d in D, p in P, c in C, r in R_lec}:
x[d,p,c,r]*s[c] <= m[r];

Room_capacity_ex{d in D, p in P, c in C, r in R_ex}:
0.8*y[d,p,c,r]*s[c]/g[c] <= m[r];

Room_capacity_com{d in D, p in P, c in C, r in R_com}:
 z[d,p,c,r]*s[c] <= m[r];

# Make sure that two courses will not be scheduled in
  the same room at the same time
```

```
Room_collision{d in D, p in P, r in R}:
sum{c in C} (x[d,p,c,r] + y[d,p,c,r] + z[d,p,c,r]) <= 1;

# Make sure that courses in the same group will not collide with eachother
Course_collision_GU1{d in D, p in P}:
sum{c in C_GU1, r in R} (x[d,p,c,r] + y[d,p,c,r]/g[c] + z[d,p,c,r]) <= 1;

Course_collision_GU2{d in D, p in P}:
sum{c in C_GU2, r in R} (x[d,p,c,r] + y[d,p,c,r]/g[c] + z[d,p,c,r]) <= 1;

Course_collision_EM1{d in D, p in P}:
sum{c in C_EM1, r in R} (x[d,p,c,r] + y[d,p,c,r]/g[c] + z[d,p,c,r]) <= 1;

Course_collision_EM2{d in D, p in P}:
sum{c in C_EM2, r in R} (x[d,p,c,r] + y[d,p,c,r]/g[c] + z[d,p,c,r]) <= 1;

Course_collision_adv{d in D, p in P}:
sum{c in C_adv, r in R} (x[d,p,c,r]) <= 1;

# Make sure that the other courses will not collide with themselves
Course_collision_others{d in D, p in P, c in C_others}:
sum{r in R} (x[d,p,c,r] + y[d,p,c,r]/g[c] + z[d,p,c,r]) <= 1;

# Make sure that the correct number of lectures, exercises,
  and computerlabs is scheduled
Lecture_sessions{c in C}:
sum{d in D, p in P, r in R_lec} x[d,p,c,r] = n_lec[c];
Excercise_sessions{c in C}:
sum{d in D, p in P, r in R_ex} y[d,p,c,r] = n_ex[c]*g[c];
Computer_sessions{c in C}:
sum{d in D, p in P, r in R_com} z[d,p,c,r] = n_com[c];

# Make sure that there is some day between lectures when possible
Sparse{d in D diff {5}, c in C diff {'MMG800','LGMA10','MMGK11','MMGL31','MMGF30'}}:
 sum{p in P, r in R} (x[d,p,c,r] + x[d+1,p,c,r]) <= 1;

# Forces the lectures for each course to be scheduled in the same room
Same_room_lec{c in C, r in R_lec}:
sum{d in D, p in P} x[d,p,c,r] - w1[c,r]*n_lec[c] = 0;

# Forces the exercises for each course to be scheduled in the same room
Same_room_ex1{c in C diff C_g, r in R_ex}:
sum{d in D, p in P} y[d,p,c,r] - w2[c,r]*n_ex[c] = 0;

Same_room_ex2{c in C_g, r1 in R_ex, r2 in R_ex}:
sum{d in D, p in P} (y[d,p,c,r1] + y[d,p,c,r2])
- w2[c,r1]*n_ex[c] - w2[c,r2]*n_ex[c] = 0;

# Forces exercises to be scheduled directly after lectures
# Works for courses that have number of lectures >= number of exercises
Ex_after_lec{d in D, p in P diff {1}, c in C diff{'MSG830','LGMA10'}}:
sum{r in R_lec} (y[d,p,c,r]/g[c] - x[d,p-1,c,r]) <= 0;

# Make sure that courses does not have more than 1 lecture each day
```

```
Lecture_limit{d in D, c in C}: sum{p in P, r in R} x[d,p,c,r] <= 1;
# The same for exercises
Excercise_limit1{d in D, c in C diff (C_g union {'MMGL31','LGMA10'})}:
 sum{p in P, r in R} y[d,p,c,r] <= 1;

Excercise_limit2{d in D}: sum{p in P, r in R} y[d,p,'MMGL31',r] <= 2;
# The same for computerlabs
Computer_limit{d in D, c in C diff {'MSG830'}}:
sum{p in P, r in R} z[d,p,c,r] <= 1;

# Locked sessions that can not be changed
MMG300: x[2,2,'MMG300','Pascal'] + y[2,3,'MMG300','MVH12']
        + x[4,2,'MMG300','Pascal'] + y[4,3,'MMG300','MVH12'] = 4;

MVG300: x[1,3,'MVG300','Euler'] + z[1,4,'MVG300','MVF22']
        + x[3,3,'MVG300','Euler']  + z[3,4,'MVG300','MVF22']
        + x[5,3,'MVG300','Euler'] + z[5,4,'MVG300','MVF22'] = 6;

LGMA10: x[2,3,'LGMA10','Pascal'] + x[4,3,'LGMA10','Pascal']
        + x[1,3,'LGMA10','Pascal'] + x[5,3,'LGMA10','Pascal']
        + y[1,1,'LGMA10','MVF31'] + y[1,2,'LGMA10','MVF31']
        + y[2,1,'LGMA10','MVF31'] + y[2,2,'LGMA10','MVF31']
        + y[4,1,'LGMA10','MVF31'] + y[4,2,'LGMA10','MVF31']
        + y[5,1,'LGMA10','MVF31'] + y[5,2,'LGMA10','MVF31'] = 12;

MSG830: x[2,2,'MSG830','Euler'] + x[4,3,'MSG830','Euler'] + y[3,2,'MSG830','MVF33']
        + z[2,3,'MSG830','MVF22'] + z[2,4,'MSG830','MVF22'] = 5;

MMGF30: x[2,2,'MMGF30','MVF23'] + x[3,3,'MMGF30','MVF23']
        + x[5,2,'MMGF30','MVF23'] = 3;
```

C Sample of MATLAB source code

The MATLAB code that was used to visualize the results for the first study period is presented below.

```
% Getting AMPL data, that is lectures (x), exercises (y) and computer labs
% (z).
Xtmp=fileread('x.txt');
Ytmp=fileread('y.txt');
Ztmp=fileread('z.txt');

% Number of courses and number of rooms
numOfCourses = 15;
numOfRooms = 11;

% Signs and words that must be deleted from the read files so that they can
% be converted into numbers
bad={'L9MA10'; 'LGMA10'; 'MMA421'; 'MMA511'; 'MMG300'; 'MMG500'; 'MMG720';
     'MMG800'; 'MMGF20'; 'MMGK11'; 'MMGL31'; 'MSA200'; 'MSG200'; 'MSG830';
     'MVG300';'Euler'; 'MVF21'; 'MVF22'; 'MVF23'; 'MVF26'; 'MVF31'; 'MVF32';
     'MVF33'; 'MVH11'; 'MVH12'; 'MMGF30';'Pascal';',';'=';':';';';'[51**]';
     '[41**]';'[31**]';'[21**]'; '[14**]';'[13**]';'[12**]';'x [11**]';
     'y [11**]';'z [11**]';'2';'3';'4';'5';'[';']';'*';'_ex';'_com'};

% Replace every element from 'bad' with an empty slot for x, y and z
for i=1:length(bad);
    Xtmp = strrep(Xtmp,bad{i,1},'');
    Ytmp = strrep(Ytmp,bad{i,1},'');
    Ztmp = strrep(Ztmp,bad{i,1},'');
end

% Convert characters to numbers
Xtmp=str2num(Xtmp);
Ytmp=str2num(Ytmp);
Ztmp=str2num(Ztmp);

% Strings with zeroes. The length 20 since we got 5 days with 4 time
% periods each
X=zeros(numOfCourses,numOfRooms,20);
Y=zeros(numOfCourses,numOfRooms,20);
Z=zeros(numOfCourses,numOfRooms,20);

% The output from AMPL is in the form of a large matrix. This matrix is
% here rewritten to a three-dimensional tensor. This is done for X, Y and Z.
% will then contain submatrices representing all the timeperiods.
for i=0:19
    X(:,:,i+1)=Xtmp(1+numOfCourses*i:numOfCourses*(i+1),:);
    Y(:,:,i+1)=Ytmp(1+numOfCourses*i:numOfCourses*(i+1),:);
    Z(:,:,i+1)=Ztmp(1+numOfCourses*i:numOfCourses*(i+1),:);
end

% Vectors with course codes for lectures, exercises and computer labs.
lectures={'\fontname{times}LGMA10 lec'; '\fontname{times}MMA421 lec';
    '\fontname{times}MMA511 lec'; '\fontname{times}MMG300 lec';
    '\fontname{times}MMG500 lec'; '\fontname{times}MMG720 lec';
    '\fontname{times}MMG800 lec'; '\fontname{times}MMGF20 lec';
    '\fontname{times}MMGF30 lec'; '\fontname{times}MMGK11 lec';
    '\fontname{times}MMGL31 lec'; '\fontname{times}MSA200 lec';
    '\fontname{times}MSG200 lec'; '\fontname{times}MSG830 lec';
    '\fontname{times}MVG300 lec'};

excercises={'\fontname{times}LGMA10 ex'; '\fontname{times}MMA421 ex';
    '\fontname{times}MMA511 ex'; '\fontname{times}MMG300 ex';
    '\fontname{times}MMG500 ex'; '\fontname{times}MMG720 ex';
    '\fontname{times}MMG800 ex'; '\fontname{times}MMGF20 ex';
    '\fontname{times}MMGF30 ex'; '\fontname{times}MMGK11 ex';
    '\fontname{times}MMGL31 ex'; '\fontname{times}MSA200 ex';
    '\fontname{times}MSG200 ex'; '\fontname{times}MSG830 ex';
    '\fontname{times}MVG300 ex'};

comlabs={'\fontname{times}LGMA10 com'; '\fontname{times}MMA421 com';
    '\fontname{times}MMA511 com'; '\fontname{times}MMG300 com';
    '\fontname{times}MMG500 com'; '\fontname{times}MMG720 com';
    '\fontname{times}MMG800 com'; '\fontname{times}MMGF20 com';
    '\fontname{times}MMGF30 com'; '\fontname{times}MMGK11 com';
    '\fontname{times}MMGL31 com'; '\fontname{times}MSA200 com';
    '\fontname{times}MSG200 com'; '\fontname{times}MSG830 com';
    '\fontname{times}MVG300 com'};

% Vector with rooms
rooms={'\fontname{times}Euler'; '\fontname{times}MVF21';
    '\fontname{times}MVF22'; '\fontname{times}MVF23'; '\fontname{times}MVF26';
    '\fontname{times}MVF31'; '\fontname{times}MVF32'; '\fontname{times}MVF33';
    '\fontname{times}MVH11'; '\fontname{times}MVH12'; '\fontname{times}Pascal'};

% Definied colors
color1 = [0,1,0];
color2 = [0.7,1,1];
color3 = [1,0.5,0.5];
color4 = [1,1,0];
%% All courses in one timetable

% Set color
coursecol=['y';'y';'y';'y';'y';'y';'y';'y';'y';'y';'y';'y';'y';'y';'y'];

figure(1)
clf
% Remove axes info
set(axes(),'XTick',[],'YTIck',[])
hold on

% Loop through all time periods
for k = 1:20

    % Choose one time period at each iteration and
    % find the non-zero indexes from this matrix
    A=sparse(X(:,:,k));

    % i and j are vectors with the indexes
```

```
[i,j]=find(A);

% Find which day we are at
day = 0.7*(ceil(k/4)-1);
% Find which period we are at
if (mod(k-1,4)+1) == 4
    period = 1.14*(mod(k-1,4)+1);
else
    period = 1.1*(mod(k-1,4)+1);
end
% Set Y displacement to zero
% It will be increased every time a session is plotted
ydisp=0;

% If we have non zero entries
if ~isempty(i)
    % A long as we have it
    for l=1:length(i)
        % Write the lecture and the room, then increase ydisp
        text(day,2*(5-period-ydisp),lectures(i(l)),'BackgroundColor',...
            coursecol(i(l)),'FontWeight','bold','FontSize',16);
        text(day+0.32,2*(5-period-ydisp),rooms(j(l)),'BackgroundColor',...
            coursecol(i(l)),'FontWeight','bold','FontSize',16);
        ydisp=ydisp+0.13;
    end
end

% Does exactly the same for exercises
A=sparse(Y(:,:,k));
[i,j]=find(A);
day = 0.7*(ceil(k/4)-1);

if (mod(k-1,4)+1) == 4
    period = 1.14*(mod(k-1,4)+1);
else
    period = 1.1*(mod(k-1,4)+1);
end

if ~isempty(i)
    for l=1:length(i)
        text(day,2*(5-period-ydisp),excercises(i(l)),'BackgroundColor',...
            coursecol(i(l)),'FontWeight','bold','FontSize',16);
        text(day+0.32,2*(5-period-ydisp),rooms(j(l)),'BackgroundColor',...
            coursecol(i(l)),'FontWeight','bold','FontSize',16);
        ydisp=ydisp+0.13;
    end
end

% Does exactly the same for computer labs
A=sparse(Z(:,:,k));
[i,j]=find(A);
day = 0.7*(ceil(k/4)-1);

if (mod(k-1,4)+1) == 4
```

```
        period = 1.14*(mod(k-1,4)+1);
    else
        period = 1.1*(mod(k-1,4)+1);
    end

    if ~isempty(i)
        for l=1:length(i)
            text(day,2*(5-period-ydisp),comlabs(i(l)),'BackgroundColor',...
                coursecol(i(l)),'FontWeight','bold','FontSize',16);
            text(day+0.32,2*(5-period-ydisp),rooms(j(l)),'BackgroundColor',...
                coursecol(i(l)),'FontWeight','bold','FontSize',16);
            ydisp=ydisp+0.13;
        end
    end
end

% Separates the time periods with lines
plot([-0.2,3.3],[8,8],'k','LineWidth',3);
plot([-0.2,3.3],[5.8,5.8],'k','LineWidth',3);
plot([-0.2,3.3],[3.6,3.6],'k','LineWidth',3);
plot([-0.2,3.3],[1.07,1.07],'k','LineWidth',3);

plot([-0.01,-0.01],[0,8.5],'k','LineWidth',3);
plot([0.575,0.575],[0,8.5],'k','LineWidth',3);
plot([1.29,1.29],[0,8.5],'k','LineWidth',3);
plot([1.99,1.99],[0,8.5],'k','LineWidth',3);
plot([2.675,2.675],[0,8.5],'k','LineWidth',3);

plot([3.29999,3.29999],[0,8.5],'k');
plot([-0.2,3.3],[8.49999,8.49999],'k');

% Draw all the days and times
text(0.16,8.2,'\fontname{times}Monday','FontSize',22,'FontWeight','bold');
text(0.8,8.2,'\fontname{times}Tuesday','FontSize',22,'FontWeight','bold');
text(1.5,8.2,'\fontname{times}Wednesday','FontSize',22,'FontWeight','bold');
text(2.2,8.2,'\fontname{times}Thursday','FontSize',22,'FontWeight','bold');
text(2.9,8.2,'\fontname{times}Friday','FontSize',22,'FontWeight','bold');

text(-0.15,7.8,'\fontname{times}8:00','FontSize',22,'FontWeight','bold');
text(-0.18,5.6,'\fontname{times}10:00','FontSize',22,'FontWeight','bold');
text(-0.18,3.4,'\fontname{times}13:15','FontSize',22,'FontWeight','bold');
text(-0.18,0.9,'\fontname{times}15:15','FontSize',22,'FontWeight','bold');

axis([-0.2 3.3 0 8.5])


%% GU1+2
% Creates a plot for only GU1 and GU2
% Most works as in the previous case
coursecol=[color2;color2;color2;color1;color2;color2;color2;color2;color2
           color2;color2;color2;color2;color2;color1];
figure(2)
clf
set(axes(),'XTick',[],'YTIck',[])
```

```
hold on

for k = 1:20

    ydisp=0;

    A=sparse(X(:,:,k));
    [i,j]=find(A);
    day = 0.7*(ceil(k/4)-1);
    period = 1.12*(mod(k-1,4)+1);

    if ~isempty(i)
        for l=1:length(i)
            % Choose only the courses we want
            if i(l)==4 || i(l)==15 || i(l)==5 || i(l)==13
                text(day,2*(5-period-ydisp),lectures(i(l)),'BackgroundColor',...
                    coursecol(i(l),:),'FontWeight','bold','FontSize',19);
                text(day+0.38,2*(5-period-ydisp),rooms(j(l)),'BackgroundColor',...
                    coursecol(i(l),:),'FontWeight','bold','FontSize',19);
                ydisp=ydisp+0.23;
            end
        end
    end

    A=sparse(Y(:,:,k));
    [i,j]=find(A);
    day = 0.7*(ceil(k/4)-1);
    period = 1.12*(mod(k-1,4)+1);

    if ~isempty(i)
        for l=1:length(i)
            if i(l)==4 || i(l)==15 || i(l)==5 || i(l)==13
                text(day,2*(5-period-ydisp),excercises(i(l)),'BackgroundColor',...
                    coursecol(i(l),:),'FontWeight','bold','FontSize',19);
                text(day+0.38,2*(5-period-ydisp),rooms(j(l)),'BackgroundColor',...
                    coursecol(i(l),:),'FontWeight','bold','FontSize',19);
                ydisp=ydisp+0.23;
            end
        end
    end

    A=sparse(Z(:,:,k));
    [i,j]=find(A);
    day = 0.7*(ceil(k/4)-1);
    period = 1.12*(mod(k-1,4)+1);

    if ~isempty(i)
        for l=1:length(i)
            if i(l)==4 || i(l)==15 || i(l)==5 || i(l)==13
                text(day,2*(5-period-ydisp),comlabs(i(l)),'BackgroundColor',...
                    coursecol(i(l),:),'FontWeight','bold','FontSize',19);
                text(day+0.38,2*(5-period-ydisp),rooms(j(l)),'BackgroundColor',...
                    coursecol(i(l),:),'FontWeight','bold','FontSize',19);
                ydisp=ydisp+0.23;
            end
        end
    end
end

plot([-0.22,3.4],[8,8],'k','LineWidth',3);
plot([-0.22,3.4],[5.85,5.85],'k','LineWidth',3);
plot([-0.22,3.4],[3.62,3.62],'k','LineWidth',3);
plot([-0.22,3.4],[1.37,1.37],'k','LineWidth',3);

plot([-0.01,-0.01],[0,8.5],'k','LineWidth',3);
plot([0.635,0.635],[0,8.5],'k','LineWidth',3);
plot([1.35,1.35],[0,8.5],'k','LineWidth',3);
plot([2.03,2.03],[0,8.5],'k','LineWidth',3);
plot([2.75,2.75],[0,8.5],'k','LineWidth',3);

plot([3.39999,3.39999],[0,8.5],'k');
plot([-0.22,3.4],[8.49999,8.49999],'k');

text(0.16,8.2,'\fontname{times}Monday','FontWeight','bold','FontSize',25);
text(0.82,8.2,'\fontname{times}Tuesday','FontWeight','bold','FontSize',25);
text(1.47,8.2,'\fontname{times}Wednesday','FontWeight','bold','FontSize',25);
text(2.2,8.2,'\fontname{times}Thursday','FontWeight','bold','FontSize',25);
text(2.94,8.2,'\fontname{times}Friday','FontWeight','bold','FontSize',25);

text(-0.18,7.7,'\fontname{times}8:00','FontWeight','bold','FontSize',25);
text(-0.22,5.55,'\fontname{times}10:00','FontWeight','bold','FontSize',25);
text(-0.22,3.35,'\fontname{times}13:15','FontWeight','bold','FontSize',25);
text(-0.22,1.05,'\fontname{times}15:15','FontWeight','bold','FontSize',25);

% Color sample
text(0.1,7.5,'\fontname{times}Color 1:','FontWeight','bold','FontSize',19);
text(0.35,7.5,'      ','BackgroundColor',color1);
text(0.1,7,'\fontname{times}Color 2:','FontWeight','bold','FontSize',19);
text(0.35,7,'      ','BackgroundColor',color2);
axis([-0.22 3.4 0 8.5])
```

Figures for Engineering mathematics and computational science, *other courses* and *advanced courses* are made in the same way as the figure for the mathematics program at the University of Gothenburg, as seen in the last cell of the code.

D Summary in Swedish

D.1 Inledning

Optimering är en gren inom matematiken som handlar om att finna den bästa möjliga lösningen till ett givet problem. Detta görs genom att uttrycka problemet som en matematisk modell. Det finns många olika områden där optimering tillämpas, några exempel är då man vill minimera kostnaden för tillverkningen av en viss produkt, eller då man vill fastställa den mest effektiva rutten för leveranser. Med dagens tekniska resurser är många problem hanterbara, men större eller mer komplexa problem är ofta så tidskrävande att resurserna ej räcker till. Därför är det ibland acceptabelt med en lösning som ligger tillräckligt nära den optimala lösningen.

Ett typiskt optimeringsproblem som uppstår inom många olika områden är schemaläggning. Ett sådant problem består av att finna ett schema som inte bara är praktiskt möjligt, utan som dessutom i största möjliga mån uppfyller önskemål från samtliga involverade parter. Ett vanligt tillvägagångssätt för att lösa denna typ av problem är heltalsoptimering.

Denna rapport beskriver modellering och lösning av ett schemaläggningsproblem för kurser som ges av Institutionen för Matematiska vetenskaper på Göteborgs universitet och Chalmers tekniska högskola, genom användning av heltalsoptimering.

Schemaläggningen på matematikinstutitionen görs just nu genom att, så långt som det är möjligt, återanvända schemat från tidigare år och manuellt göra de ändringar som krävs. Detta är en väldigt tidskrävande process som kan ta flera veckor att slutföra. En datoriserad metod skulle göra denna process betydligt snabbare och mindre ansträngande.

D.2 Specifikation av schemläggningsproblemet

Vid utformningen av ett schema måste hänsyn tas till underliggande regler som avgör huruvida två händelser får ske samtidigt. Detta antyder att det finns vissa egenskaper som ett schema måste uppfylla. Det svåra ligger i att placera en viss kurs i ett visst rum vid en viss tid med en viss lärare, utan att den krockar med någon annan kurs. Vissa kurser får lov att ges samtidigt i olika rum, medan andra ej får ges samtidigt på grund av att studenter läser flera kurser åt gången. Det finns ett bestämt antal lektionstimmar för varje kurs som måste schemaläggas varje vecka, samtidigt som varje kurs måste vara schemalagd i ett passande rum.

Förutom villkor som måste uppfyllas finns personliga åsikter från lärare och elever om vad som utgör ett bra schema. Dessa åsikter bör också tas i åtanke vid utformningen av schemat.

Schemaläggningen utförs för våren 2013 som består av två läsperioder, där vardera läsperiod sträcker sig över åtta veckor. Antalet kurser som ska schemaläggas är 15 för första läsperioden respektive 13 för den andra. Det finns 11 rum och antalet studenter varierar mellan 4 och 68 för de olika kurserna. Det resulterande schemat ska uppfylla alla nödvändiga krav för att schemat ska vara praktiskt möjligt. Dessutom ska önskemål från lärare och studenter uppfyllas i största möjlliga mån.

D.3 Teori

Den matematiska modellen till ett optimeringsproblem består av en målfunktion, som ska minimeras eller maximeras, samt bivillkor. Det finns två typer av bivillkor: hårda bivillkor, som måste uppfyllas för att lösningen ska vara praktiskt möjlig, och mjuka bivillkor, som helst ska uppfyllas.

Det finns olika områden inom optimering för att uttrycka olika typer av problem. Exempel på sådana områden är linjär- och heltalsoptimering. Ett linjärt problem består endast av linjära bivillkor och en linjär målfunktion. För sådana problem finns effektiva metoder och de är relativt lättlösta. Den mest använda metoden är simplexmetoden. Ett linjärt problem där alla variabler dessutom är heltal kallas för heltalsproblem. I den här studien modelleras problemet som ett binärt problem där variablerna endast antar värdena 0 och 1.

Många optimeringsproblem kräver ansenlig beräkningskraft. För att lösa dessa kan det matematiska modelleringsspråket AMPL användas tillsammans med en lösare vilken i standardfallet är CPLEX. AMPL är ett effektivt verktyg vid lösning av optimeringsproblem, då syntaxen i AMPL liknar den matematiska notation som används inom optimering. För att lösa ett problem i AMPL behövs två filer, en modell-fil innhållande den matematiska modellen och en data-fil innehållande all nödvändig data. AMPL anropar sedan en lösare, till exempel CPLEX, som utnyttjar olika metoder för att lösa det givna problemet. Då det gäller heltalsproblem använder CPLEX branch-and-cut vilken är en kombination av branch-and-bound och plansnittningsmetoder. I branch-and-bound-metoden delas problemet upp i mindre, mer lättlösta problem. Anledningen till att de är mer lättlösta är att heltalskravet tas bort och därför kan de lösas som linjära problem. På varje delproblem tillämpas plansnittning, det vill säga nya bivillkor läggs till för att begränsa den tillåtna lösningsmängden.

D.4 Metod

För att formulera den matematiska modellen krävdes förståelse för huvudkoncepten inom optimeringsteori. Då förkunskaperna inom optimering varierade mellan gruppens medlemmar gjordes först individuella litteraturstudier för att samtliga medlemmar skulle få tillräcklig teoretisk förståelse för att kunna delta i modelleringen.

Nödvändig information om kurser, rum och lärare inhämtades från personer ansvariga för schemaläggningen på matematikinstitutionen. Denna information kompletterades med information hämtad från kurshemsidor.

Den matematiska modellen formulerades och implementerades i AMPL, där problemet sedan löstes med CPLEX. För att visualisera resultatet användes MATLAB. Modellen undersöktes till sist med känslighetsanalys för att avgöra modellens stabilitet och hur variabler och olika bivillkor påverkar lösningen.

D.5 Modell

För att uttrycka problemet matematiskt används binära variabler. Detta är lämpligt eftersom en variabel ska representera svaret på frågan huruvida en kurs ska läggas på en viss tid eller inte. Variablerna har fyra index som anger dag, tid, kurs och rum. Om kurs c schemaläggs under dag d, på tid p, i rum r så är $x_{d,p,c,r} = 1$. Annars är $x_{d,p,c,r} = 0$. Modellen beskriver således ett heltalsoptimeringsproblem. Målfunktionen är utformad så att den fokuserar schemat till mitten av dagarna. Detta görs genom att lägga vikter på tidiga och sena föreläsningar. Detta innebär att variablerna för de icke önskvärda tiderna multipliceras med en faktor större än 0 vilket gör att målfunktionen antar ett större värde om dessa tidpunkter används. Då målfunktionen ska minimeras innebär detta att de oönskade tidpunkterna undviks.

Kurserna som schemaläggs är uppdelade i olika grupper, beroende på vilka kurser som får lov att schemaläggas samtidigt. De olika grupperna är:

- Kurser för studenter på Göteborgs universitets matematikprogram år ett.
- Kurser för studenter på Göteborgs universitets matematikprogram år två.

- Kurser för studenter på Engineering mathematics and computational science på Chalmers tekniska högskola år ett.
- Kurser för studenter på Engineering mathematics and computational science på Chalmers tekniska högskola år två.
- *Avancerade kurser.*
- *Andra kurser.*

Avancerade kurser är kurser för de som går tredje året på matematikprogrammet och *andra kurser* är de kurser som inte ingår i någon annan grupp.

Bivillkoren som behövs i modellen är formulerade så att schemat ska bli praktiskt möjligt samtidigt som det ska uppfylla så många önskemål som möjligt. De hårda bivillkoren, som måste vara uppfyllda, är:

1. Det får inte vara fler än en föreläsning, räkneövning eller datorövning i ett visst rum vid en viss tid.
2. Schemat måste vara fullständigt, det vill säga alla kurser måste schemaläggas med rätt antal lektioner.
3. Rummen måste vara tillräckligt stora för kurserna.
4. Kurser får inte krocka med sig själva.
5. Kollisioner mellan kurser inom samma grupp är inte tillåtet, förutom för *avancerade kurser* och *andra kurser.*
6. Föreläsningar i kurser i gruppen *avancerad kurser* får inte krocka.
7. Föreläsningar, räkneövningar och datorövningar måste ges i rätt typ av rum.

De mjuka bivillkoren, som helst ska uppfyllas, ges av:

1. I en kurs bör det inte vara fler än en föreläsning, en räkneövning och en datorövning per dag.
2. Schemat för varje grupp av studenter bör vara så utspritt som möjligt över veckan och så kompakt som möjligt under dagen.
3. Räkneövningar bör komma direkt efter tillhörande föreläsning.
4. Varje föreläsning i en kurs bör ges i samma rum. Detsamma gäller för räkneövningar.
5. Måndag morgon och fredag eftermiddag bör inte innehålla några lektioner.
6. Lektioner bör helst ligga mitt på dagen, det vill säga undvik tidiga morgnar och sena eftermiddagar.

D.6 Känslighetsanalys

Den känslighetsanalys som utförts gäller främst hur modellen reagerar på förändringar hos bivillkor och målfunktion. Det finns begränsat med teori för känslighetsanalys till heltalsproblem, men genom att tillfälligt ta bort vissa bivillkor kan man få en uppfattning om hur de påverkar lösningen. Om ett hårt bivillkor tas bort blir resultatet ett ogiltigt schema vilket är ointressant att analysera. Därför avlägsnas endast de mjuka bivillkoren.

Den största begränsningen i modellen är antalet kurser som ej får krocka med varandra. Om alla kurser får ges samtidigt skulle det vara möjligt att schemalägga 220 lektioner varje vecka (5 dagar, 4 tidsperioder, 11 rum) men med kurser som inte får ges samtidigt minskar detta antal avsevärt.

Det resulterande schemat påverkas mycket av förändringar i målfunktionen, då vikterna kan ändras så att schemat koncentreras till de tider som är önskvärda. De vikter som är valda i den nuvarande målfunktionen är de som bäst uppfyller lärares och elevers önskemål.

D.7 Resultat och slutsats

Det genom modellen framtagna schemat uppfyller alla hårda bivillkor vilket garanterar att det är praktiskt möjligt. Schemat är dessutom bra i den mening att det uppfyller de mjuka bivillkoren i högre grad än det nuvarande schemat. Den formulerade modellen är tillräckligt generell för att kunna användas för andra läsperioder och kurser och genom att ändra vikterna i målfunktionen kan modellen anpassas till nya önskemål. Lösningar skulle behöva analyseras av de som i slutändan ansvarar för schemaläggningen men arbetsbördan skulle minska avsevärt.

I nuläget kräver alla modifikationer av modellen grundläggande förståelse för AMPL och MATLAB och utförlig kunskap om modellen. En möjlig fortsättning för att vidare underlätta schemaläggningen på matematikinstitutionen vore att utveckla ett lättförståeligt användargränssnitt. Detta skulle göra det möjligt att göra ändringar och omedelbart se det nya schemat.