Dynamic Room Allocation

Adaptive planning of teaching facilities at the University of Twente

Master thesis
May 1st 2015

Assessment committee: Prof.dr. Marc J. Uetz
Rudy A. Oude Vrielink MSc. BA
Dr.ir. Gerhard F. Post
Prof.dr.ir. Bart L.J.M. Nieuwenhuis

Author: Dorien F.T.F. Meijer Cluwen BSc.

UNIVERSITY OF TWENTE.

Documentation Page

Title	Dynamic Room Allocation
Subtitle	Adaptive planning of teaching facilities at the University of Twente
Author	F.T.F. Meijer Cluwen BSc. University of Twente Discrete Mathematics and Mathematical Programming f.t.f.meijercluwen@alumnus.utwente.nl
Supervisors	Prof.dr. M.J. Uetz University of Twente Discrete Mathematics and Mathematical Programming m.j.uetz@utwente.nl R.A. Oude Vrielink MSc.BA University of Twente Centre for Educational Support r.a.oudevrielink@utwente.nl
Keywords	room allocation, timetabling, automatic dynamic, classroom allocation problem
Date of publication	May 1, 2015
Cover art	IITO Group 7 [Alan, Bakker, Muis, van de Pieterman, and Teerenstra, 2015]

Abstract

Creating timetables for an educational institution is a complex problem. Especially if that institution consists of over 9000 students, 1500 teachers, 55 curricula, 250 rooms and 60.000 educational events a year. Furthermore it also includes a lot of stakeholders, each with their own interests. A timetable is prone to frequent modifications, which can be anything from cancelled lessons to renovating locations and adding new studies. Hence it has a direct impact on the functionality of the university and vice versa.

The University of Twente is struggling to plan all educational events given the availability of educational facilities and teachers. It seems that lecture rooms are used suboptimal; utilisation of one room might be low, while another lecture might be cancelled due to lack of space. Sometimes a lecture room is empty while, according to the timetable, it should be occupied or visa versa. Currently, timetables are made before the first-year students have registered and the current students have enrolled for courses. It is therefore difficult to estimate the required capacity of a given course.

To decrease uncertainty on actual utilisation, research has been done on ways to count student numbers during lectures. The recent interlinking of different student information systems might prove new insights in the student enrolment. However, the question is if and how this information on enrolment and participation can improve the quality of the timetable.

Because of complexity, only the allocation of scheduled events to classrooms will be researched, it is assumed that a schedule in time is given and fixed.

The goal of this research is to analyse the effect of usage and utilisation knowledge on the quality of the room allocation. This is done in two steps; creating a room allocation automatically and using this model to dynamically adapt to real-time data.

The room allocations made in this research are based on, and compared to, the actual allocation of the first quartile of the year 2014-2015 at the UT. An initial allocation will be made using Mathematical Programming after which the quartile is simulated using Discrete Event Simulation. Information on estimated student enrolment and attendance is known and used to simulate changes in utilisation and occupation.

Samenvatting

Een rooster creëren voor een onderwijsinstelling is een hele opgave. Zeker als die instelling bestaat uit meer dan 9000 studenten, 1500 docenten, 55 curricula, 250 zalen en 60.000 onderwijs activiteiten per jaar. Het probleem bevat tevens een hoop verschillende betrokken partijen, elk met hun eigen belangen. Ook wordt een rooster vaak gewijzigd, dit kan alles zijn van uitgevallen colleges tot het verbouwen van locaties en toevoegen van nieuwe studies. Een rooster heeft dus direct impact op de functionaliteit van de universiteit en andersom.

De Universiteit Twente worstelt met het plannen van al haar onderwijs activiteiten gegeven de beschikbare faciliteiten en docenten. Het lijkt alsof zalen sub-optimaal gebruikt worden; de benutting van sommige zalen kan laag zijn, terwijl er colleges niet door kunnen gaan omdat daarvoor geen ruimte beschikbaar is. Soms staan zalen leeg terwijl er wel iets in gepland was of andersom. Momenteel worden de roosters gemaakt voordat de eerstejaars studenten geregistreerd zijn en de huidige studenten ingeschreven zijn voor hun vakken. Het is daarom moeilijk om een inschatting te doen van de benodigde capaciteit van een vak.

Om de onzekerheid over de daadwerkelijke benutting te verkleinen is er onderzoek gedaan naar manieren om het aantal studenten te tellen tijdens colleges. En de recente koppeling van verschillende student-informatiesystemen kunnen nieuwe inzichten geven in de studentenaantallen. De vraag is echter of en hoe deze informatie over inschrijvingen en deelname aan vakken de kwaliteit van het rooster kunnen verbeteren.

Vanwege de complexiteit wordt in dit onderzoek enkel gekeken naar de toewijzing van activiteiten aan zalen en niet aan tijd. Er wordt aangenomen dat tijden waarop activiteiten gepland zijn vast staan en al bekend zijn.

Het doel van dit onderzoek is om te analyseren wat het effect is van het bezettings- en benuttings-informatie op de kwaliteit van de zaalallocatie. Dit wordt gedaan in twee stappen; het automatisch creëren van een allocatie en het gebruiken van dit model om dynamisch te plannen met behulp van realtime data.

De zaalallocaties in dit onderzoek zijn gebaseerd op, en worden vergeleken met, de werkelijke allocatie in het eerste kwartiel in 2014-2015 op de UT. De initiële allocatie wordt gemaakt met behulp van Mathematisch Programmeren, waarna het kwartiel wordt gesimuleerd met behulp van Discrete Event Simulatie. Informatie over geschatte studenten inschrijvingen en deelname is bekend en wordt gebruikt om veranderingen in utilisatie en occupatie te simuleren.

Preface

This thesis is the capstone of my Masters study Applied Mathematics at the University of Twente (UT). It was conducted at the chair of Discrete Mathematics and Mathematical Programming (DMMP) and in association with the Centre of Educational Support.

I was always a big fan of sliding puzzles, riddles and 24-games. I even used to request math equations for dessert. That I decided to study Math was no surprise to my family. And even though I still cannot solve a Rubik's cube, because I refuse to look up the trick on the internet, I still love to solve riddles. Good thing too, as a Master Thesis is a riddle in and of itself.

After working on the mathematics behind Smart Grids during my bachelor thesis and internship I decided I would like to do my final project in a completely different direction. Via the CHOIR (Centre for Healthcare Operations Improvement and Research) research group I ended up at a new research direction of Erwin Hans, namely the 'Project Timetable Innovation' which he started in conjunction with Marc Uetz and Rudy Oude Vrielink. I was instantly sold and the PATAT conference (Practice and Theory of Automatic Timetabling) made me realise that this was a direction of research which packed together all the things I love about math.

This thesis would not be here without the help and support of a lot of people. Firstly I would like to thank my two supervisors, Marc Uetz and Rudy Oude Vrielink, for their unlimited patience every time I told them I had it almost working. I would like to thank Marc for his helpful comments and for always having confidence that everything would work out OK. I would like to thank Rudy for his never ending stream of ideas, which in itself could fill a complete chapter. And for reading and rereading my report over and over.

I would like to thank Gerhard Post for joining my assessment committee and for the random half an hour in the bus that we talked about decomposition techniques. You made my day, or more correctly the Quartile. Also a big thanks to Bart Nieuwenhuis who, last minute, completed my assessment committee.

My thanks also go to the scheduler team at the UT, and to Bas Hoekstra for reading my report and minimising the amount of untruths I naively put in there about the scheduling process at the UT. I thank the people from DMMP and SOR (and whoever decided to join as well) for the companionable coffee and lunch breaks. I thank my friends and family for their support and interest, and for sometimes not asking when I would finally graduate. Last but not least I thank Melle for his moral support when needed.

Dorien F.T.F. Meijer Cluwen,
✉ f.t.f.meijercluwen@alumnus.utwente.nl
Enschede, May 2015

> *"The essence of mathematics is not to make simple things complicated, but to make complicated things simple."*
>
> - S. Gudder

Contents

Chapter 1

Problem Description

This chapter gives an overview of the problem at hand. Section 1.1 clarifies the relevance of the problem, Section 1.2 gives a short overview of the current literature and in Section 1.3 the goals and approach of this research are explained and an outline of the thesis is given.

1.1 Introduction

The university of Twente (UT) has recently implemented a new educational model. This new educational model (called "Twente Education Model" or TEM) consists of thematic modules, independent educational units consisting of several subjects and projects on the same theme. The increase of projects and tutor sessions increase the demand for smaller educational facilities, while large introductory courses (like the mathematical course for all technical faculties) increase the demand for large lecture rooms. This increases the complexity of allocating rooms and the schedulers at the UT struggle to plan all courses given the availability of educational facilities and teachers. Barely enough space was available during the introduction of TEM and it is estimated that the situation will become more dire in the next few years.

Currently timetables are made before the first-year students have registered and the current students have enrolled for courses. It is therefore difficult to estimate the required capacity of a course. To make sure the capacity meets the demand, the schedulers often reserve extra space (also called slack or white space). Which may lead to under-use of some rooms, while large events are allocated to rooms which are too small or are not allocated at all. Other examples of incomplete information can be; teacher availability, room requirements and not reporting when a room is underused or not used at all. Which all can lead to an inefficient use of rooms.

Apart from educational events rooms can also be reserved for self study, meetings, congresses, symposia, etc. These reservations are made anywhere between a few minutes to years beforehand. The facility management would like to honour as much of them as possible. Incomplete information about room usage and utilisation might lead to less available rooms for reservations and hence a loss of profits from external rental of rooms.

More capacity can be gained by acquiring extra space, planning lectures during evening hours and or hiring more teachers. One could also change the layout of current rooms to make them more flexible, i.e. suitable for different types of educational events.

Apart from more varied rooms the new educational system also requires more time. Student timetables have few free hours. When not following a lecture or tutorial students are expected, and scheduled, to do self study or projects. Hence the task of creating (clash free) student and teacher timetables (allocation to both room and time) is a complex one. Therefore the scheduler team has less (or no) time to improve the timetable and do things like decreasing the number of different rooms used for a course. I.e. the quality of the timetable decreases.

The use of algorithms to automatically produce timetables and or room allocations can lessen the load of the schedulers. Not all demands and wishes are however easy to translate to computer language and just like the scheduler teams work the output of an algorithm is dependent on the input. If student enrolment is badly estimated then a room allocation algorithm will not provide a suitable allocation. And neither will a room allocation be usable if student attendance deviates significantly from student enrolment.

The relevance of this problem can be seen from Figure 1.2 and 1.1. During the 6^{th} week of term the average utilisation of all used rooms is almost 50% lower than planned. Meanwhile more rooms are in use than originally planned. The higher occupation can partly be explained by the fact that free rooms are used by students for self study and unsupervised projects. Only Monday and Friday in the late afternoon the occupation is lower than planned. Students seem to prefer education during the middle of the day and the middle of the week.
The low utilisation could indicate two things; not all enrolled students are attending or student enrolment was estimated (and thus planned) to high.

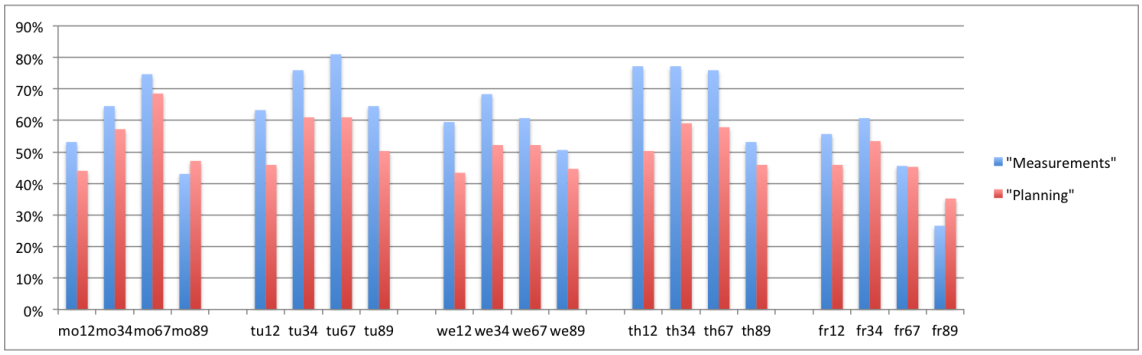


Figure 1.1: Average occupation over all rooms per time slots during week 41

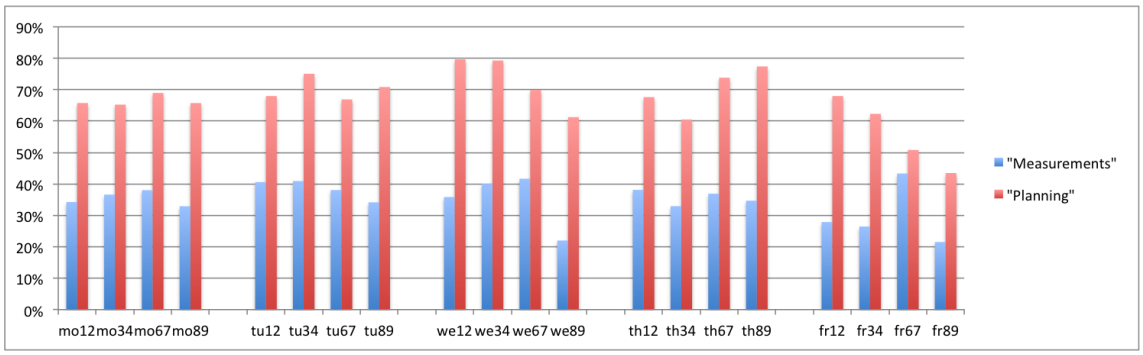


Figure 1.2: Average utilisation over all rooms per time slots during week 41

To decrease uncertainty on actual utilisation research has been done on ways to count student numbers during lectures (Baars, van den Berg, and Höner [2014]) and the recent interlinking of different student information systems might also prove new insights in the student enrolment. The question is however if this information on enrolment and participation can improve the quality of the timetable. Because of complexity, only the allocation of scheduled events to classrooms will be researched, it is assumed that a schedule in time is given and fixed.

The goal of this research is to analyse the effect of usage and utilisation knowledge on the quality of the room allocation. This is done in two steps; creating a room allocation automatically and using this model to dynamically adapt to real-time data.

The room allocations made in this research are based on and compared to the actual timetable of the first quartile of the year 2014-2015 at the UT. A schedule will be made using Mathematical Programming after which the quartile is simulated using Discrete Event Simulation. Information on estimated student enrolment and attendance is known and used to simulate changes in utilisation and occupation.

1.2 Background

The allocation of educational events (e.g. lectures, tutorials, workshops, recitals) to available room space in order to satisfy as many requirements and constraints as possible, is known in literature as the Classroom Assignment Problem (CAP) or Teaching Space Allocation Problem. The complexity of this problem depends on the constraints and objectives, which might differ greatly between different institutions or even faculties.

In its simplest form, when all events have the same duration, each time slot can be solved as an independent assignment problem. This problem is equivalent to finding a maximum weighted bipartite matching between the events and the rooms, where the edge weights represent the degree in which a room satisfies an event (see Figure 1.3). The Hungarian algorithm solves this problem in polynomial time.

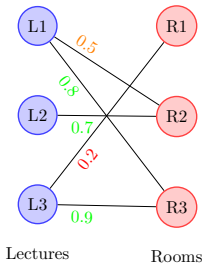


Figure 1.3: Classroom assignment expressed as weighted bipartite graph

In reality events have different durations and should (in most cases) only occupy one room for the entirety of their duration (also called *contiguous room stability*). This leads to interdependency between different time slots, Carter and Tovey [1992] proved that this problem is NP-hard even for just two time slots. Even more complex formulations of the classroom assignment problem also take into account constraints which cause interdependencies between non-contiguous time slots, like minimising the number of different rooms used by a certain course on a certain day, week or even semester (also called *course room stability*).

Constraints can also cause interdependencies between rooms, in which case one cannot decompose the problem per group of rooms. This can happen when taking into account different types of events and rooms, of which some cannot be combined (like a lecture in a workplace) and some event types can use the same room-type.
Other constraints, which do not necessarily make the problem easier or harder, are for example; the number of rooms occupied at a certain time or day, under-use or overuse— of the capacity of a room (*space wastage*), rooms with special facilities (*room requirements*), proximity to department/previous lesson/etc. (*proximity*).

The CAP is part of the much larger and well studied (University) Course Timetabling Problem (CTP), in which times and resources (like rooms, teachers and students) are assigned to events so as to satisfy a set of given constraints as much as possible. A survey of approaches to the timetabling problem up until 2010 is done among others by Schaerf [1999], Carter and Laporte [1998], Burke and Petrovic [2002] and MirHassani and Habibi [2013]. There is also a biannual

conference on Practice and Theory of Automated Timetabling (PATAT) which provides a bridge between researchers and practitioners, and which proceedings contain most recent research on the Course Timetabling Problem and its sub problems.

The approaches to solving the CTP range from simple constructive heuristics to intricate algorithms which combine multiple techniques from among others Operation Research and Artificial Intelligence. MirHassani and Habibi divide the approaches into model based methods, like mathematical programming, and heuristic methods. The latter is again divided in four subtypes; sequential methods, clustering methods, constraint-based methods and meta-heuristic methods.

Sequential methods were one of the first methods to be used. In the early days the approaches were based on the human way of solving the problem; lectures are scheduled in order of decreasing difficulty, where the difficulty of scheduling depends on the chosen rules and quality measures. When conflicts arise events can be swapped to another time or room so as to make room for other events. These methods are also called constructive heuristics, because they start with an empty timetable and have the goal to construct a feasible solution.

Later on, researchers applied more general techniques to solving the timetabling problem. By representing the timetabling problem as a graph one could use graph colouring or network flow heuristics. In graph colouring each node corresponds to an event, each colour to a time slot (or room) and edges denote that two events cannot occur at the same time (or place). By ensuring that no node has the same colour as its neighbour, events joined by an edge will not occur at the same time (nor in the same room).
There are also methods to improve a given timetable. A lot of research is done on the use of techniques from artificial intelligence. Most common are Simulated Annealing, Tabu Search, Genetic Algorithms and hybrid approaches. Other widely used approaches are logic programming and constraint programming.

Due to the increase of computer power and research into computational optimisation techniques, integer programming (IP) methods have become more popular. While two decades ago classical IP techniques could only solve models of at most a hundred integer variables, today models with many thousands of binary variables pose no problem. However the timetabling problem is a very large and complex, hence there still is need for techniques to reduce the solution space. The timetabling problem is therefore often decomposed into the allocation of time slots and the allocation of rooms (see for example Burke, Mareček, Parkes, and H.Rudová [2010]). Because the first problem takes a lot more resources and constraints into account it is usually more interesting and more profitable to research. That might explain why a lot of literature on University Course Timetabling can be found but only a handful on Classroom Assignment.

Related problems in academic institutions to the CAP are among others Exam Room Assignment and Office Space Allocation. While all three try to find an allocation which appoints a room to each event, exam or person, their requirements

and constraints can differ greatly. For example, in Office Space Allocation and Exam Room Allocation one would like to group several persons/exams in one room (to increase the utilisation and decrease the number of needed rooms), while it is not desirable or not even allowed to put several lectures in one room.
In Burke and Varley [1998] an analysis was done on space allocation in 96 British universities. Emphasis was placed on how large and diverse the problem was in each university, what computer tools where used (if any) and which constraints the university imposed. They concluded that the problem varied greatly among the institutions, some constraints were common, whereas other in direct conflict with each other. Many of the universities used computers, but only a few used computerised automation of space allocation. Other examples of research on (Office) Space Allocation are found in Burke and Varley [1999], Burke et al. [2001], Landa-Silva [2003], they all provide heuristic methods to allocate space while trying to satisfy constraints as; waste and overuse of space, room specific requirements (like access for disabled persons) and proximity requirements. Constraints which are also useful for the CAP.
Another useful constraint can be found in Ayob and Malik [2011], who describe an Examination-Room Assignment Problem which aims to minimise students moving between rooms which have consecutive examinations. The paper by Elloumi et al. [2014] also describes exam timetabling but provides claims and proves which are useful for the CAP.

When looking specifically to CAP one of the most relevant and cited paper is written by Carter and Tovey [1992] and discusses, as its title indicates, the complexity of the Classroom Assignment Problem in different circumstances. Their focus is on the kind of cases that schedulers might encounter in practice and explains why room assignment is sometimes easy and sometimes immensely difficult. Note that computationally easy problems will still take weeks or even months when solved by hand.

Approaches to solving this problem are varied; from (meta-) heuristics to constraint logic programming to integer programming. An overview of obtained literature on solving approaches for CAP is given in Table 1.1. The papers by Abdennadher et al. [2000], Constantino et al. [2010], Martinez-Alfaro and Flores-Teran [1998], Martinez-Alfaro et al. [1996], Phillips et al. [2015] are abbreviated to [M96], [M98], [C10], [A00] and [P15].
The approach in this research will be to use (mixed) integer programming and the used constraints will follow from Section 2.1 and are further explained in Section 2.2.

Even when a problem is feasible it might still be difficult to find a schedule which utilises the rooms efficiently. In Beyrouthy et al. [2007b, 2010] some explanations are given why the utilisation of teaching space in Universities is notoriously low. It has all to do with how good the rooms and events match, a big room is not always a suitable room. The first paper of Beyrouthy et al. gives methods to visualise and demonstrate the mismatch between the event and room-size profiles, while the latter investigates the type of rooms and events and their mismatch. Room and event types play a crucial role in this research as well, the education at the UT is diverse and requirements of rooms differ between types of events, or even among events of the same type.

Table 1.1: Literature on approaches to solve CAP

paper	approach	constraints/objectives
[M96]	simulated annealing	one class per room at a time one class for an instructor at a time minimise deviation from required room requirements preferred meeting time of instructor
[M98]	simulated annealing	minimise deviation from room capacity minimise distance between classroom and its department fill buildings from bottom to top
[C10]	iterate heuristic & variable neighbourhood search	only one lesson in a room at the same time (except for special lectures) do not exceed the room capacity concentrate classes in the same course within a given area assign classes according to class year (lower numbered rooms are for freshman) assign all meetings of a class to the same room (weekly)
[A00]	constraint logic programming	only one course in a room at the same time minimise deviation from room capacity assign event to a specific room assign event to a specific building assign event to a room with specific equipment
[P15]	integer programming & maximum set packing	maximise number of assigned events maximise the number of hours spent by students in all allocated events (used when not all events can be assigned)

Another reason that might explain low utilisation is the deviation of student enrolment versus its estimation and the variable nature of student attendance during a scheduling horizon. As timetables are often (as in this research) made before student enrolment is allowed, a timetable which was good originally might not be suitable when real enrolment is known. There is also a difference between enrolled students and students who attend an educational event. The general idea is that student attendance decreases through the term and increases again when tests approach.

When more data is known on student enrolment and attendance, better estimates can be made. This leads to a more robust timetable. When there is not enough information to produce good estimates, as is the case in this research (at least for student attendance), timetabling and or room allocation should be done more dynamically in order to deal with this uncertainty.

Literature on dynamic classroom allocation could not be found, let alone those in which different time horizons are compared. Dynamic allocation is however extensively examined in e.g traffic networks, computer networks and telephone networks. This research is therefore not only relevant to the University of Twente but can also extend the literature on dynamic assignment of classrooms.

1.3 Research goals and approach

Making timetables for an educational institution is a complex problem. Especially if that institution consists of over 9000 students, 1500 teachers, 55 curricula, 250 rooms[1] and 60.000 educational events a year. Furthermore it also includes a lot of stakeholders, each with their own interests. A timetable is prone to frequent modifications, which can be anything from cancelled lessons to rebuilding locations and adding new studies. Hence it has a direct impact on the functionality of the university and vice versa.

Because of complexity, only the allocation of scheduled events to classrooms will be researched, it is assumed that a schedule in time is given and fixed. Such a decomposition corresponds to the scheduling process at the University of Twente (UT), i.e. the schedule in time is largely made before rooms are allocated to the events. An overview of the scheduling process at the UT is given in **Chapter 2**.
The goal of this research is to create a model which can utilise real time data, like occupation and utilisation, to dynamically allocate rooms. This model should incorporate wishes and demands of the stakeholders at the UT. It could be used to simulate the effect of these wishes and demands on the room allocation. E.g. what happens to utilisation and occupation of rooms when room suitability is taken into account with more/less priority.

The model is also used to compare the effect of real time data on the quality of the room allocation for different scheduling horizons. This way an insight can be gained into the usefulness of real time data and whether and when (re)allocation should be done. It is hypothesised, from Figure 1.2 and 1.1, that reacting to real time data could indeed improve utilisation and occupation of rooms. When utilisation is lower than expected smaller rooms could be allocated to the events and less energy would be consumed to heat the rooms. Or money could be earned by renting the large rooms to external parties.

The research goal is divided in the following two sub goals;

I Automatically produce an acceptable concept room allocation (instead of scheduling by hand)

II Increase the quality of the room allocation by dynamic room assignment using enrolment- and real-time data

The question directly arises what an acceptable room allocation is and how the quality of an allocation can be measured. This depends on stakeholders and their wishes and demands regarding room allocation, which are described in **Chapter 2**. This chapter also gives an overview of the timetabling process at the UT and how this will be incorporated in the model. The model architecture denotes how and when the previously described sub goals are used. The sub goals themselves are discussed in **Chapters 3 - 6**.

[1] Including project rooms

Chapter 3 Presents an Integer Linear Programming model to create a classroom allocation for a given time horizon. Which is used to answer questions like; Can we find a feasible room allocation? Can we approximate the actual schedule in Quartile 1A? Can we obtain an average occupation of 70%? And what does this do to the other measures? Can we schedule a room empty / what happens when a room cannot be used during a certain period of time? Questions like these are answered in **Chapter 4**.

Chapter 5 shows how Discrete event simulation is used to simulate the student enrolment and participation. And **Chapter 6** uses this technique to show if the current amount of reservations can be honoured and if so, what would happen if the amount of reservations increases? This chapter also shows if the automatically produced allocation of sub goal I can honour more reservations than a manual allocation.

The back matter of this report consists of a conclusion (see **Chapter 7**), some appendices and an overview of the used references.

Chapter 2

Model Architecture

This chapter describes the current situation with regard to timetabling at the University of Twente and how this will be incorporated in this research. Section 2.1 gives an overview of the educational structure at the UT and how a timetable is constructed. This leads to a set of constraints and wishes, which are described as key performance indicators in Section 2.2. In Section 2.3 a short recap of the CAP is given and notations are given for needed input and output. This chapter ends with a description of the model architecture used for this problem, which explains how the room allocation process of the UT is modelled and which adaptations are needed to take real-time data into account.

2.1 Timetabling at the University of Twente

An educational year at the UT is split up in two semesters, each of which is split into two quartiles. The summer holiday can also be seen as a semester and is mostly used to plan resits and summer courses. Because this period is fairly quiet it will be omitted from this research. A quartile consists of 10 weeks, excluding holidays. In the old education system the last two weeks were reserved for exams and resits, in the new education system exams are given anywhere during the quartile and the last two weeks are often reserved for projects.

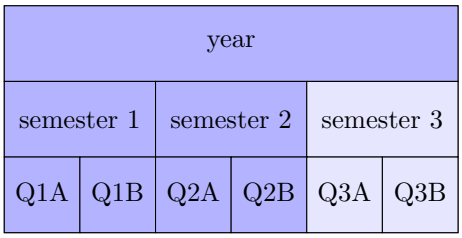


Figure 2.1: Structure of an educational year

Currently the UT has two educational systems,the new educational model has been introduced to the first two years of all Bachelor studies and the third year for some pilot studies. The educational system for Master students has not changed. The structure of both systems is given in Figure 2.2, common is that programmes are part of studies, which are governed by faculties. In the new system each yearly programme consists of four modules, each given in a separate quartile. Each module consists of several module sections, which revolve around a certain theme or topic.

In that sense a module section looks equivalent to a course, one of the differences lies in passing or failing a module section. A module cannot (officially) be passed partly, either a student passes a module or fails it. In practice these rules are often less strict.

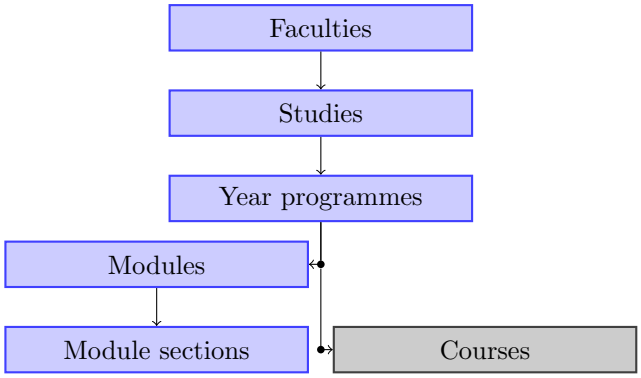


Figure 2.2: Structure of the educational system

Another difference to the previous system is the flexibility of the timetable, i.e. how much freedom the schedulers still have to change the start time and/or date of an event. While previously a teacher only had to compose its own course, they now have to take the other module sections and their teachers into account. A single teacher has much more flexibility and can deliver a proposed course schedule which is still very flexible with respect to time. Nowadays some module teams produce a schedule for a whole module, in which almost all time slots (during which education is normally given, i.e. 8:45-17:30 at the UT) are occupied. This leads to a less flexible and hence more complex scheduling problem. Partly this is due to the modules requiring (almost) all contact hours, but also because no good protocols for the new system exist yet. Note that the flexibility of allocating rooms has not changed, teachers can still propose a number of requirements which the schedulers will gratify if possible.

In the year 2012 the UT has changed its timetabling process drastically. Until then the schedulers were situated inside the faculties themselves and were tasked with scheduling their faculty. This had the advantage of being close to the educational staff. The schedulers would request their rooms through a centralised booking system, which was managed by the reservation bureau (RB). The room requirements they could indicate were very limited and room allocation was done by people who had less knowledge on education. It also took RB around two weeks to provide the schedulers with a room allocation, which they now do themselves in around two days.

Nowadays the schedulers are located in the same office and work as a team. A specialised timetabling software is used, Syllabus+ (Scientia Ltd.). This software maintains a database of all events and can easily spot inconsistencies as double bookings of a room, teacher or student(group). It also gives the possibility to automatically assign time slots and rooms, the first is, as indicated previously, not very flexible at the UT and the second seems to work suboptimal. The latter could indicate that the used heuristics are suboptimal, but also that some constraints are not or incorrectly incorporated. This is not an unrealistic assumption, as it is hard to describe constraints in computer language and some constraints are just

not supported by the current version of the software at the UT.
Syllabus+ also has many additional options, like exam scheduling and student allocation, but these are costly. Before deciding to purchase an optional module for the software it is advisable to do research on the benefits they could provide and if these outweigh the costs.

The timetabling process is formally defined in the protocol educational planning [Punt et al., 2012], the timeline given in this document (and further elaborated in [Dijksterhuis, 2014]) is given in Table 2.1.

Table 2.1: Timeline of the timetabling process for the scheduler team, Quartile 1 (Sep-Nov)

timeline	activity
< Mar	Scheduling or reserving rooms on fixed times for orations, promotions, symposia
< Mar	Scheduling large events in fixed rooms (if approved)
Mar - mid Apr	Receiving timetable information forms (non-TEM)
Mar - mid Apr	Verifying timetable information forms
> Apr 9^{th}	Creating module sections in Syllabus+ and courses for non-TEM
Feb - May	Receiving timetable information TEM
Apr - May	Creating activities (events) in Syllabus+
Apr - May	Scheduling and room allocation in Syllabus+
mid June	Sending concept timetable to module coordinators
mid June	Receiving feedback from module coordinators
end June	Updating concept timetable
end June	Sending updated concept timetable to teachers etc.
mid July	Publishing final timetable
<= Nov	Rescheduling events (on demand)

When a schedule is created, a certain order of scheduling is followed. First priority is given to the most constrained events, e.g. those which are so big they can only be placed in the biggest room, events which need to be recorded or tutorial groups which should occur close to each other. After these highly constrained events are allocated the rest of the events are allocated in order of decreasing size.

In Dijksterhuis [2014] an analysis of the bottlenecks of Table 2.1 is given. He notes that;

1. Deadlines are not always met.
2. Information on education needs to be assembled very early.
3. Received educational information is non-uniform.
4. Stakeholders take more freedom than granted.
5. Room allocation cannot use final enrolment data.
6. Feedback is slow.
7. The current sequence of the process might not be optimal.

The first three bottlenecks are mainly about communication and organisation. Information on deadlines does not seem to reach the right persons or they do not seem to understand the consequences of it. To decrease the time needed to correctly assemble education information good communication is needed between the module coordinators and the scheduler team, better communication will also lead to an information input which is more suitable for the scheduler software and therefore easier to implement.

The fourth bottleneck occurs because it is often unclear what stakeholders can and cannot demand. For example, a module team goes creative in their creation of their new module. They proudly present their innovative and optimal timetable, unaware that it breaks several norms and some of the rooms they claim would be more suitable for other events. If a protocol is made which clarifies which stakeholders can demand what, a fairer timetable could be made.

Bottleneck number five is due to the the fact that room allocation is done before the final enrolment is done. Since a few years there is a law which indicates that students whom pre-enrol before May 1^{st} are entitled to a 'study check'[1]. This is an activity in which the student and the educational institute learn more about each other. In this way the government hopes to reduce the number of dropouts and students that switch studies. For some studies the check is compulsory and a way to filter possible drop-outs beforehand. Actual enrolment needs to be done before September 1^{st}. Course enrolment at the UT can be done from 2 months in advance up until the first week of the corresponding Quartile starts[2].

When the scheduler team starts with creating the timetable, neither course enrolment nor actual study enrolment is known. Therefore the estimated student enrolment is based on the number of pre-enrolled students (for freshman) and the number of enrolled students of the previous cohort. When the actual student course enrolment is known only the bottlenecks, those events which do not fit any more in their assigned room, are reallocated.
Bottleneck six leads to delay in the schedule process. The problem is that teachers can only criticise the concept timetable when it is finished and the scheduler team can only finalise the timetable when the teachers have criticised it.

In the last bottleneck the author wonders if the order of allocation might lead to better timetables and less bottlenecks. E.g. allocate teachers to courses at the end of the process instead of at the start. This way the teachers would not have to state their availability months beforehand and hopefully produce this information faster (before the deadline) at the end of the process. Note however that teachers are responsible for the content of a course and therefore need to enough time to prepare a course.

[1] http://www.rijksoverheid.nl/onderwerpen/hoger-onderwijs/studiekeuze-en-toelating
[2] http://www.utwente.nl/ces/studentservices/osiris/schema-inschrijving.pdf

2.2 Key Performance Indicators

The definition of a good room allocation is part of the question of what defines a good timetable. This definition differs per stakeholder and is often contradictory, e.g. a faculty might prefer to spread education over the week while students might prefer clustered education. Both Westerik [2013] and Dijksterhuis [2014] describe several Key Performance Indicators (KPIs) which are indicated by stakeholders from the University of Twente (UT) to measure the performance of a timetable at the UT. Note that these KPIs (when generalised) can also be used for other universities and educational institutes.

Stakeholders with respect to timetabling at the UT are; the scheduler team, students, teachers, module coordinators, educational coordinators, directors of studies, facility management and the booking office. Dijksterhuis [2014] has inventoried the interests of these stakeholders by interviewing them, this has led to the requirements given in Table 2.2. Note that only a select subset of each group of stakeholders is interviewed. Other students, teachers, etc. might provide different, even conflicting, requirements.

Table 2.2: Stakeholder requirements on room allocation (from Dijksterhuis [2014])

Students
◇ Preference to have events, which are repeated weekly, in the same building (but not necessarily the same room).
◇ Modification of rooms is allowed, provided that communication happens timely.
Educational coordinators
◇ Preference to have events, which are repeated weekly, in the same room.
◇ minimise the number of room modifications.
◇ Education of a study should happen in the same building.
◇ Projects should be scheduled in close proximity of each other.
◇ Take into account events that should not be close to each other.
Directors of studies
◇ A room must be suitable for the types of the allocated events.
◇ Rooms for self study or projects should be near the teachers office.
Facility management
◇ Make optimal use of rooms, both in utilisation and occupation.
◇ Minimise the number of rooms.

The requirements can be grouped in the following constraints;

- suitability constraints
- course room stability constraints
- occupation constraints
- utilisation constraints
- proximity constraints
- deviation constraints

Suitability constraints concern the suitability of a room for a given event, this can depend on the type of room but also on the equipment available in the room. Course room stability constraints have to do with the number of different rooms used for a course during a given period. Occupation and utilisation constraints deal with the usage of rooms. Proximity constraints cover any constraint that has to do with distances, from those between two parallel tutorials to the distance a student or teacher should travel at most on a day. Deviation constraints relate to the number and or impact of deviations from a given timetable.

This research will focus on the following KPIs; feasibility, room suitability, deviation, utilisation, occupation and course room stability. Note that proximity constraints are not taken into account, collecting the correct information and needed constraint is left as a recommendation.

2.2.1 Feasibility

The feasibility constraint measures if a room allocation is executable. A feasible room allocation is assumed to adhere the following assumptions, note that in most cases the data can be manipulated to represent the same data in a different way such that it can satisfy these assumptions.

Assumption 1. *One room per event.*

When an event needs multiple rooms, we can split them into different events. With a parameter we can keep track of these events, this allows us for example to plan events near each other. With the use of algorithms tutorials can be split up into different student groups of a certain size, depending on the available rooms [Beyrouthy et al., 2007a]. A parameter which tells if an event can be split or not needs to be introduced as well.

Assumption 2. *At most one event per room.*

When multiple events need to occur in one room, we can combine them into one event. A parameter could be introduced to keep track of which events can be combined and which cannot. Or, in the case of multiple students in a project room, one could decide to block the room for other events and reserve it for those events, while not actually scheduling the events themselves.

Assumption 3. *The event should fit in the room.*

This assumption means that a room allocated to an event should have enough space to accommodate that event. This could also be seen as a preference, i.e. a maximum utilisation of the room. The capacity of a room is defined in number of seats, which can depend on the type of activity and fire safety regulations.

Assumption 4. *An event always stays in the same room.*

When an event occurs which needs to move to another room halfway through the event duration one can split the event in multiple events which do stay in the same room. E.g. when a teacher first gives a lecture and then indicates the students to move to the computer lab, these can be split into separate events. A parameter could be created to keep track of the distance in space and/or time of these separate events.

Assumption 5. *Some rooms cannot be used at the same time.*

The UT uses virtual rooms to represent the exam-version of a room, an exam version often has less capacity (to reduce cheating). Obviously the actual room and its exam-version cannot be used at the same time. The same assumption can also be used to make sure that a combined room and the actual rooms cannot be used at the same time.

Assumption 6. *Only exams can make use of exam-rooms.*

Because exam-rooms often have less capacity than their real counterpart the model will prioritise exam-versions over normal rooms when minimising utilisation. To prevent this from happening non-exam events are not allowed in exam-rooms.

These assumptions lead to the following KPIs;

KPI 1 (Allocations) The percentage of events that is allocated

KPI 2 (Misfits) The number of events that are allocated to a room that is too small

Double bookings and moving an event halfway through their duration are excluded from this research.

2.2.2 Room suitability

At the UT room suitability is measured in two ways; by room-type and by room equipment. Rooms of the same type often satisfy the same room requirements, i.e. a room type can be seen as a set of room requirements. Room suitability will be measured by the percentage of room requirements that are met. Note that one could see the size of an event as a requirement as well, which could then be given priority over other requirements or not.

KPI 3 (REQ) The percentage of room requirements that are met.

2.2.3 Deviation

Deviation arises when modifications need to be made on a given room allocation, if this is unwanted behaviour depends on the moment in time and the stakeholder.
E.g. (according to Table 2.2) the students do not mind room modifications as long as they are communicated timely. If the complete allocation is changed before term and this is communicated beforehand this is no problem. Modifications of room allocation a day beforehand are however unwanted.
Teachers however might see this differently. When they request a different room during term, because the allocated room is not suitable any more, they want a modification to the current room allocation.

KPI 4 (DEV) The percentage of room-event pairs that differ (i.e. are modified) compared to a given room allocation.

An example of deviation can be the number events allocated to a different room after course enrolment is known (compared to the allocation based on the estimated enrolment). One would like to reallocate events which have significantly more or less students than estimated, but not (all of) those which have not changed significantly. Another example is the deviation between the allocation made before term has started and the one during term. When changes occur one would like to dynamically reallocate some events and not all events.

2.2.4 Utilisation

There are many definitions of utilisation and occupation, in this research utilisation will denote the proportion of capacity in a room that is used, while occupation will denote the proportion of time the room is used.

Definition 1. (Room utilisation) The proportion of the available *capacity* of the room that is used during a given time horizon.

Note that the UT has rooms of very different sizes, the smallest project room has size 1, while the capacity of the largest lecture room is almost 900. If utilisation is averaged among all rooms the size of a room should be taken into account, otherwise a half empty room of size 2 is just as bad as a half empty room of size 900. Two KPIs are chosen to measure the utilisation, the first denotes the average utilisation over all used rooms, the second measures unused and overused seats *space wastage*.

KPI 5 (UTLn) The average utilisation of all used rooms (with size of at least n) during a given time horizon.

KPI 6 (SPACE) The percentage of unused/overused seats in occupied rooms during a given time horizon.

Here UTL looks at the utilisation over all used rooms and, for instance, UTL20 looks at the utilisation of all used rooms with size of at least 20.

2.2.5 Occupation

Occupation refers to the time a room is occupied. As it is assumed that each event is allocated to a room, it is not helpful to count the number of used rooms per time slots. The chosen KPIs therefore looks at the number of used rooms during a day.

Definition 2. (Room occupancy) The proportion of the available *time* the room is occupied during a given time horizon.

KPI 7 (OCCn) The proportion of rooms (with size of at least n) that have been occupied during a given time horizon.

KPI 8 (ROOM) The number of rooms that have been occupied at least once during a given time horizon

Examples of OCCn are OCC100 which only looks at rooms with size of at least 100 and OCC which looks at all rooms.

2.2.6 Course room stability

Course room stability refers to the number of rooms used for a given course, and possibly the pattern in which they are used. E.g. it could be preferable to use room A for the first few weeks and room B for the rest, instead of alternately having to use room A and B. Because the suitability of a room depends on the type of an event, and a course almost always has different types of events (e.g. lectures, workshops etc.) it is illogical to minimise the number of rooms per course. This research therefore defines *course-types*, a course has a course-type if it has events of that type. Now we can minimise the number of different rooms per course-type.

KPI 9 (CROOM) The number of different rooms that are used for a given course-type.

2.3 Classroom Allocation Problem

The allocation of educational events (e.g. lectures, tutorials, workshops, recitals) to available room space in order to satisfy as many requirements and constraints as possible, is known in literature as the Classroom Assignment Problem (CAP) or Teaching Space Allocation Problem. The complexity of this problem depends on the constraints and objectives, which might differ greatly between different institutions or even faculties.

The CAP can be formulated as an Integer Linear Program (ILP) and is based on the Assignment Problem (see Figure 1.3). An ILP consists of three parts; decision variables, demands & wishes and an objective function. In this section the CAP problem will be described in words, for a mathematical notation and more information on the constraints used in this research see Section 3.1.

Table 2.3 gives an overview of the resources used in this research. Other applicable resources are among others; programmes, buildings, teachers and students. These are not (directly) used in this research but can be added in subsequent research to make use of constraints as proximity constraints and constraints to minimise the number of used buildings (especially useful during the evenings). Note that 'Course' is used to describe a set of corresponding events, i.e. it describes both courses of the old system and module sections of the new system. This can be done because room requirements are requested on event level, i.e. requirements can differ per event.

Table 2.3: Resources for the CAP

Time slot	Time interval of predefined (fixed) length
Room	Teaching space
Event	Scheduled meeting of students and/or teachers
Course	Set of events on the same topic, usually followed by the same students and given by the same teacher
Properties	Equipment in a room or the type of a room
Time horizon	Set of time slots, e.g. day, week, quartile, semester

The needed input should give information on all these resources. The most import characteristics of an event are;

- Size: (estimated) number of students at the event.
- Time slots: time slots during which the event is scheduled.
- Type: sort of event, e.g. lecture, workshop, project.
- Requirements: type of room or room equipment required by an event.

For rooms these are:

- Capacity: maximum number of students/seats in the room.
- Time slots: time slots during which the room is available for allocation.
- Type: type of the room, e.g. lecture room, workshop room, project room etc.
- Properties: the equipment a room has, e.g. whiteboard, computer.

Table 2.4 defines the CAP problem used in this research in words, with the demands and wishes based on the assumptions and KPIs defined in Section 2.2. The problem is feasible when all demands are satisfied, which corresponds to assumptions 1 to 3, 5 and 6. Note that assumption 4 is satisfied automatically as soon as each event is allocated to exactly one room.

Table 2.4: Formulation of CAP in words

Decision variables
Do I allocate room r to event e?
Demands
◇ Allocate exactly one room to each event.
◇ Allocate at most one event to a room at the same time.
◇ An event should fit in the allocated room.
◇ Some rooms cannot be used at the same time.
◇ Only exams can use exam-rooms.
Wishes
◇ Allocate an event to a room which satisfies the given requirements.
◇ Ensure a minimum utilisation per room.
◇ Ensure a minimum average occupation per used room, over a given time horizon.
◇ Minimize the number of different rooms per course-type.
◇ Minimize the deviation from a given initial allocation.
◇ Minimize the under-/overuse of seats in a room.
Objective function
Maximise the quality of the room allocation.

A solution is optimal when it is feasible and the wishes are satisfied as well as possible, the importance of each wish is denoted in the objective function. In this research a penalty is given to each wish which is not satisfied and the objective function is a weighted summation of these penalties. By adapting the weight we can change the priority and importance of each wish.

The output of a CAP is the best found room allocation for the given input and used penalty weights. The KPIs from Section 2.2 corresponding to this room allocation, give the quality of the room allocation, from several perspectives.

2.4 Model architecture

The model architecture uses the framework for Education Planning and Control by Dijksterhuis [2014], which is based on the framework for Healthcare Planning and Control of Hans, Van Houdenhoven, and Hulshof [2011]. This framework, given in Figure 2.3, splits the timetabling process into different management areas and management levels. Each management area controls the education planning in a different way. IT Planning concerns itself with the used software systems, Education Planning with the development of education, Human Resource Planning with the staff, Facility Planning with rooms and timetables and Financial Planning with money.

	Educational Planning	**Human Resource Planning**	**IT Planning**	**Financial Planning**	**Facility Planning**
Strategic	Development vision & policy: Strategy & long term plans. E.g. UT2020 & new educational models (TEM)	Personnel policies, selection procedures, University Teaching Qualification (BKO)	Set-up of IT-architecture: what functions must be delivered and what systems are needed?	Financial allocation model, budget planning, investment plans	Manage long term capacity needed for education
Tactical	Protocols. E.g. Teaching and Examination Regulations (TER), Educational information and planning	(Base) qualification for education, training and development of staff	Improvement of functionality, updates of software systems	Allocation of costs and budgets to departments and/or faculties	Drawing up year plans and semester *timetables*. Manage temporary extra capacity, e.g. evening openings/teaching
Offline Operational	Planning of education. E.g. modules, laboratories, etc.	Match teachers and other staff(e.g. tutors) to education	Use of software systems by teachers, students and staff	Clearing of costs, taxations, collections	Allocation of rooms to schedule = *Room Allocation*
Online Operational	Monitoring and correcting for unwanted effects	Monitoring and correcting for unwanted effects	Monitoring and correcting for unwanted effects	Monitoring and correcting for unwanted effects. E.g. not reporting a cancelled reservation costs money	Monitoring and correcting for unwanted effects. E.g. switch rooms when a beamer is defect. Measure performance.

Figure 2.3: Framework Education Planning & Control

The management levels split the decision process in different time horizons, each horizon has its own time span and corresponding decisions. In the strategic level long term decisions are made, these decisions are based on forecast and usually span several years. Strategic level decisions have a lot of impact. The tactical level consist of medium term decisions and spans several weeks or months. More detailed information is known compared to the strategic level, but a tactical decision has less impact than a strategic decision.

Operational decisions are made when demand is fixed, these are also called 'last minute' changes. The offline operational level acts according to decisions which are made beforehand, while the online operational level monitors the process and reacts to (unexpected) conditions.

Room allocation is part of the Facility Planning area and spans the tactical, offline operational and online operational management levels (see the light blue blocks). The dynamic allocation takes place during the online operational level (see the dark blue block). This research focusses on the tactical and operational level, although advice for the strategical level is also possible. E.g. if a structural shortage on rooms of a specific type is spotted, advice could be to arrange more of these rooms.

The proposed model architecture for (dynamic) room allocation is given in Figure 2.4. Each level has its own input, goal, approach and output. Several months in advance a room allocation is made based on a schedule with fixed times and with pre-enrolment data. In this research it is assumed that a schedule with fixed times/dates (but not rooms) exists and that start times cannot be changed any more. In future research this can be adapted, in that case shifting in time should still be possible on the tactical level.

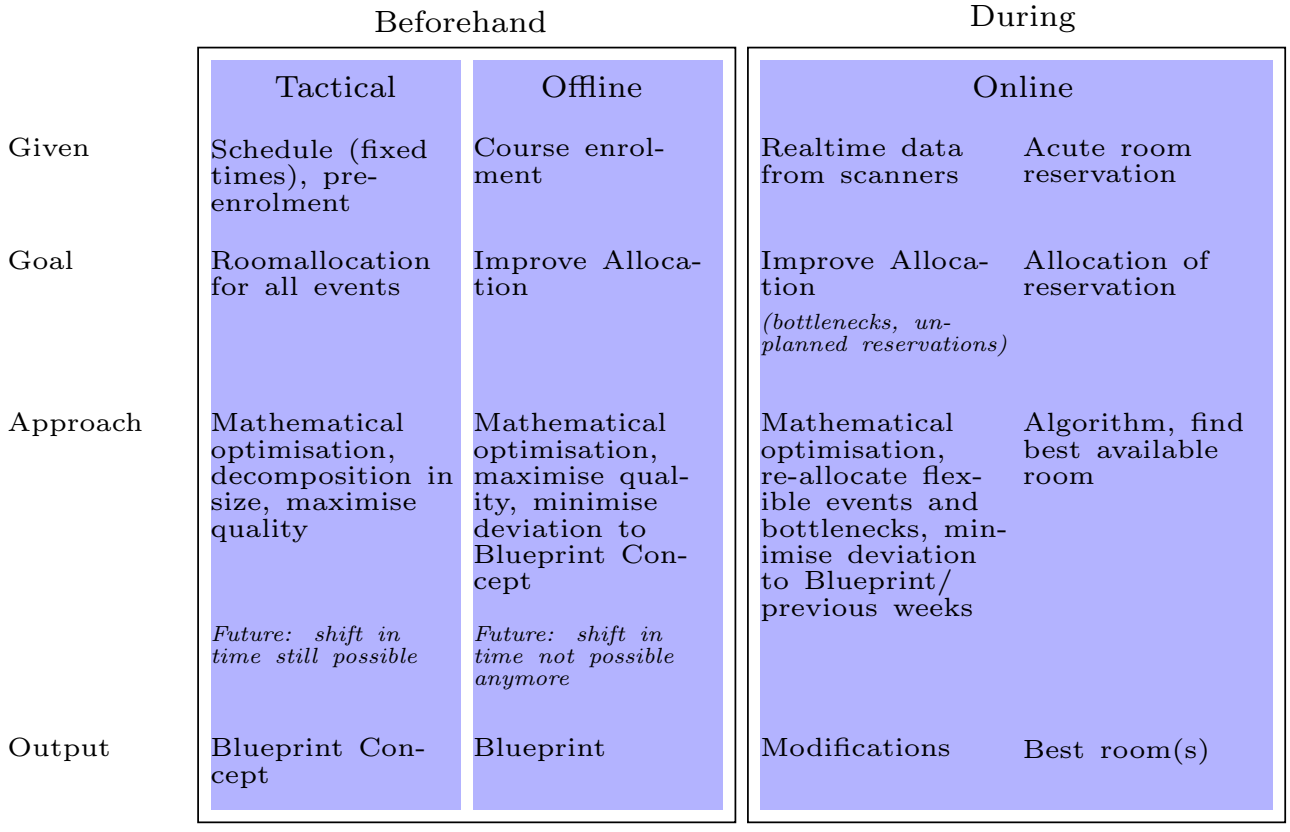


Figure 2.4: Proposed model architecture

The first sub goal of this research is to provide an automatic room allocation for a complete Quartile, this yields a quite complex problem which needs a lot of computer power. Decomposition of the problem into sub problems can be done to decrease the size and thus needed computer power to solve the problem. More information about decomposition can be found in Section 3.2.

As soon as the real course enrolment data is known modifications should be made for courses which have more or much less students than expected. Because this happens only a few weeks before the term starts it is not preferable to shift in time (see Table 2.2: Stakeholder requirements on room allocation) and deviations compared to the concept allocation should not be to large.

During the term manual counting and/or sensors can detect bottlenecks, this allows the model to adapt the future room allocation. Improving the allocation should not be done constantly, otherwise the allocation would vary through time too much. I.e. this could be done each day or week.
To preserve constraints like course room stability information on previously used and in the future allocated rooms need to be taken into account. Hence both the future allocation, as given by the Blueprint, and the past allocation (Blueprint with modifications) need to be taken into account when reallocating events.

When improving the allocation it is not preferable to reallocate every event, hence only a subset of events should be chosen to re-allocate. This also has the benefit of reducing the complexity of the allocation. Note that large events are more difficult the re-allocate, both because of the limited number of large rooms and the large number of people (read: opinions). The same can be said for exams and experiments, these kind of events are defined non-flexible events.
Only flexible events, and of course the bottleneck events, will be taken into account for online allocation improvement. More information on this decomposition is given in subsection 3.2.2.

During term it can also happen that an acute reservation is requested, this is a reservation (which could contain anything from small projects to big lectures) which needs a room immediately. A reservation can be made between two weeks before up until a few minutes beforehand. The algorithm for acute reservations is done online, if a room is available that satisfies the given requirements the reservation is allocated to that room. If no such room is available the algorithm returns a list of proposed rooms which differ either in size, properties or is available during another time or date.

The mathematical optimisation refers to the ILP model from Section 2.3 and Chapter 3. The course enrolment data, scanner data and reservations will be simulated in Chapter 5 with the use of Discrete Event Simulation, this chapter also includes the algorithm to find the best available room for an acute reservation.

Automatic room allocation

Chapter 3

Mathematical model

This chapter describes the approach to create an automatic room allocation. Section 3.1 describes the used Integer Linear Program (ILP) used to allocate rooms to events. The set of constraints and wishes from Chapter 2 are translated to hard constraints (subsection 3.1.1) and soft constraints (subsection 3.1.2). The soft constraints are used to form an objective function, which is described in subsection 3.1.3. Section 3.2 proposes two ways to reduce the complexity of the CAP, which will be used in this report.

3.1 Description of the ILP model

In this section a description is given of a mathematical model to allocate rooms to educational events. This model must decide which room to allocate to which event, while taking objectives and constraints from Chapter 2 into account.

Constraints can be divided in two types; hard constraints and soft constraints. Hard constraints (feasibility constraints or demands) set conditions that are required to be satisfied, a solution not satisfying all hard constraints is called 'infeasible'. Soft constraints (preference constraints or wishes) set conditions which are preferred, the degree in which a solution satisfies the soft constraints determines the value of that solution. A solution which is both feasible and satisfies all soft constraints (or there is no other solution which satisfies them better) is called an optimal solution.

In this research the events have already been allocated to a specific date and starting time, but no rooms have been assigned yet. Our decision variables should therefore indicate if a certain event is allocated to a certain room, i.e;

$$x_{e,r} = \begin{cases} 1, & \text{if event } e \text{ is allocated to room } r \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Note that, from the perspective of rooms, we still have to take time into account. Otherwise only one event could be allocated to a room per scheduling horizon (day, week, quartile etc.). Taking time into account for events as well is not needed, as these are already fixed in time.

We therefore denote E_t as the set of events which occur during time slot t. The length of a time slot is chosen as the greatest common divisor of the duration of all events. In this way each event is scheduled during exactly one or multiple time slots, in our case this results in time slots of 15 minutes.

3.1.1 Hard constraints

Using the decision variables we can now define hard constraints, as given in Table 2.4;

H1 Allocate exactly one room to each event.

H2 Allocate at most one event to a room at the same time.

H3 An event should fit in the allocated room.

H4 Some rooms cannot be used at the same time.

H5 Only exams can use of exam-rooms.

Note that the possibility of splitting or combining events is not yet taken into account.

H1: Allocate exactly one room to each event
The constraint of one room per event can be described by saying that out of the list $(x_{e,r1}, x_{e,r2}, ...)$ for a certain event e exactly one of them equals 1. Because the variables $x_{e,r}$ in formula 3.1 can only take the values 0 or 1, this is equivalent to saying that the sum of the variables in the list $(x_{e,r1}, x_{e,r2}, ...)$ should equal one.

$$\sum_r x_{e,r} = 1 \quad \text{for all } e \tag{H1}$$

The average of all these constraints gives us the percentage of events that are allocated, i.e. KPI 1. Note that in a feasible solution all events are allocated and hence this KPI measures 100%, i.e. it is a measure of infeasibility.

$$Allocations = \frac{\sum_{e,r} x_{e,r}}{|E|} \tag{KPI 1}$$

Where $|E|$ denotes the number of events in the problem.

H2: Allocate at most one event to a room at the same time
In the same way we can define the constraint which restricts the number of events per room to at most one. This time however, we have to take time into account, if we do not do this we can only use a room once per scheduling horizon. As the $x_{e,r}$ variables only directly influence events and rooms but not time, time is not taken into account directly. However, this can be done indirectly by using $e \in E_t$ (the events occurring during slot t). This set is known as a schedule in time is already given.

The constraint can now be described by saying that at most one variable out of the list $(x_{e1,r}, x_{e2,r}, ...)$ for each room r and each time slot t can equal 1, where $e1, e2, .. \in E_t$ are the events in time slot t. This is equivalent to saying that the sum of variables in this list should be at most one.

$$\sum_{e \in E_t} x_{e,r} \leq 1 \quad \text{for all } r, t \tag{H2}$$

H3: An event should fit in the allocated room
Let cap_e be the number of people in event e and cap_r the number of people which fit in room r, then assumption 3 can be modelled by;

$$x_{e,r} cap_e \leq cap_r \quad \text{for all } e, r \tag{H3}$$

The number of events that are allocated to a too small room (KPI 2) can be obtained by counting the number of H3 constraints which are unsatisfied, i.e. this KPI is also a measure of infeasibility. When the model is solved this number is easily computed by checking for each allocated event-room pair if the size of the event cap_e exceeds the size of the room cap_r.

$$Misfits = \#\text{unsatisfied H3 constraints} \tag{KPI 2}$$

H4: Some rooms cannot be used at the same time.
Let $conflict_r$ be the set of rooms that conflict with room r, i.e. cannot be used at the same time as room r. Then assumption 4 can be modelled as followed

$$\sum_{e,r1 \in conflict_r} x_{e,r} + x_{e,r_1} \leq 1 \quad \text{for all } e, r \tag{H4}$$

H5: Only exams can make use of exam-rooms.
Because exam-rooms often have less capacity than their real counterpart the model will prioritise exam-versions over normal rooms when minimising utilisation. To prevent the model from placing non-exam events in exam-rooms the decision variables for those event-room combinations are set to zero. Let E_{exam} be the set of events which have an exam related type and R_{exam} rooms which have an exam related type.

$$x_{e,r} = 0 \quad \text{for all } e \notin E_{exam}, r \in R_{exam} \tag{H5}$$

3.1.2 Soft constraints

When a solution satisfies the above constraints it is said to be feasible, but not necessarily optimal. When allocating a room to an event there are also preferences and wishes of stakeholders to adhere to (see Table 2.4). These preference constraints (soft constraints) each have their own weight, which might be very high. These weights are part of the objective function and determine the importance of each constraint compared to the other constraints. The soft constraints used in this report are;

S1 Allocate an event to a room which satisfies the given requirements.

S2 Ensure a minimum utilisation per room.

S3 Ensure a minimum average occupation per used room (per day).

S4 minimise the number of different rooms per course-type.

S5 minimise the deviation of a given initial allocation.

S6 minimise the space wastage.

A soft constraint is made by relaxing a hard constraint, i.e. allowing the constraint to be unsatisfied but penalising this. The hard constraint $Ax \leq b$ can be softened by adding the penalty variable ϵ;

$$Ax \leq b \Longrightarrow Ax \leq b + \epsilon$$

When the constraint is unsatisfied the penalty variable is forced to take a positive value. By adding the penalty variable to the objective function one can minimise the penalty variable, and hence the deviation from the aspired hard constraint. I.e. the penalty variable denotes the degree of dissatisfaction. Each soft constraint in this report is given its own penalty variable (denoted in gray) and penalty weight. This penalty weight denotes the importance of that constraint (compared to the other soft constraints).

Depending on the soft constraint a penalty variable can adopt values in different ranges.

- **The constraint is either satisfied or unsatisfied.**
  For example soft constraint S1, either all requirements are satisfied or not. In this case the penalty variable is either zero or one. These variables are denoted by a $y \in \{0, 1\}$.
- **The constraint can be unsatisfied by a certain percentage.**
  For example soft constraint S2, the utilisation can deviate with a certain percentage from the required minimum utilisation. In this case the penalty variable denotes this deviation and can be at most 100%. These variables are denoted by a $z \in (0, 1)$.

- **The constraint can be unsatisfied by a certain number.**
  For example soft constraint S6, each unused or overused seat (space wastage) is a deviation from the aspired zero under-/overuse. In this case the penalty variable can take a finite positive integer number, this number is finite as the number of seats and students are finite. The used penalty variables of this type are $rooms_{ctype}$ and $sw_{e,r}$.

S1: Allocate an event to a room which satisfies the given requirements
Let $require_{e,req}$ equal one if event e requires requirement req (e.g. the need for a beamer or a lecture room) and zero otherwise. Let $sat_{r,req}$ equal one if room r satisfies requirement req and zero otherwise. Let $y_{e,req}$ be a binary variable that indicates if the preference constraint is satisfied for event e and requirement req (0) or unsatisfied (1). The requirement constraints can then be modelled by;

$$\sum_r (x_{e,r} require_{e,req}) \leq y_{e,req} + \sum_r (x_{e,r} sat_{r,req}) \quad \text{for all } e, req \tag{S1}$$

The average over all these penalty variables defines the percentage of room requirements that could not be met. Hence the suitability (percentage that can be met, KPI 3) can be calculated by;

$$REQ = 1 - \frac{\sum_{e,req} y_{e,req}}{|E||req|} \tag{KPI 3}$$

Here $|E|$ denotes the number of events and $|req|$ the number of possible room requirements. Note that this measure gives the aggregated suitability over all requirements, one could also calculate the suitability of each separate requirement by only summing over that specific requirement and not dividing by $|req|$.

S2: Ensure a minimum utilisation per room.
By setting a minimum utilisation we can match event and room sizes better and therefore save the big rooms for events that might arrive after the room allocation is made or for existing events that need to be rescheduled. Let $\underline{utl}_r$ denote the minimum utilisation for room r (in %) and $z_{e,r}$ an integer variable between zero and one that indicates the degree to which the constraint is unsatisfied, i.e. when the constraint is satisfied $z_{e,r}$ equals zero and the more the constraint becomes unsatisfied the larger $z_{e,r}$ becomes. We can now describe the minimum utilisation constraints by;

$$utl_{e,r} + z_{e,r} \geq x_{e,r} \underline{utl}_r \quad \text{for all } e, r \tag{S2}$$

Here the utilisation of a room r given an event e is given by;

$$utl_{e,r} = \frac{cap_e}{cap_r}$$

To calculate the average utilisation of all used rooms during a given time horizon T (KPI 5) the utilisation of a room per time slot t and time horizon T are needed. The utilisation of a room r during a time slot t depends on the event allocated to room r. If an event e is allocated to room r $x_{e,r}$ equals one, hence $x_{e,r}utl_{e,r}$ denotes the utilisation at slot t. If an event e is not allocated to room r $x_{e,r}$ equals zero and so does $x_{e,r}utl_{e,r}$. As exactly one event can be allocated to room r summing over all possible events (read: only those which occur in time slot t, i.e. events in E_t) gives us the utilisation of room r during slot t.

$$utl_{r,t} = \sum_{e \in E_t} x_{e,r} utl_{e,r}$$

The utilisation of room r during time horizon T is weighted over the slots in which it is actually occupied. At those slots the occupation of room r during slot t ($occ_{r,t}$) has value one.

$$utl_{r,T} = \frac{\sum_{t \in T} utl_{r,t} occ_{r,t}}{\sum_{t \in T} occ_{r,t}} \tag{3.2}$$

Now KPI 5 can be described as the weighted average of the average utilisation of each room, the weights equal the capacities such that bigger rooms have more influence than smaller rooms. Here the horizon T coincides with the length of the scheduling horizon.

$$UTLn = \frac{\sum_{r \in R_n} utl_{r,T} cap_r}{\sum_{r \in R_n} cap_r} \tag{KPI 5}$$

Where R_n denotes the set of rooms with size of at least n.

S3: Ensure a minimum average occupation per used room (per day).
Aside from the proportion of used capacity which is used, one can also look at the time during which a room is used, the average occupation of a room. This can be done for a whole semester, but also for each quartile, month, week, day etc. Let T be the time horizon over which the average occupation is calculated, i.e. a sequence of $|T|$ time slots t. Let $z_{r,T}$ be an integer variable that indicates the degree to which the constraint is unsatisfied for room r and time horizon T. And let $\underline{occ}_{r,T}$ the minimum average occupation for room r (in %) during time horizon T. Note that the constraint is satisfied by default if the room is deliberately not used during horizon T at all (costs could be saved on heating, electricity, cleaning etc.). Hence either;

$$occ_{r,T} = 0$$

or

$$occ_{r,T} + z_{r,T} \geq \underline{occ}_{r,T}$$

Where $occ_{r,T}$ denotes the average occupancy of room r during the time horizon T.

These OR-constraints can be modelled using a binary variable $\alpha_{r,T}$. If $\alpha_{r,T}$ equals 1 then the second constraint (equation S3b) forces the average occupation to zero (the room r is not used during T), if $\alpha_{r,T}$ equals 0 the room can be used and a penalty is given when the minimum average occupation is not met.

$$\begin{array}{lllll} \text{a)} & occ_{r,T} + z_{r,T} + \alpha_{r,T} & \geq & \underline{occ}_{r,T} & \text{for all } r, T \\ \text{b)} & occ_{r,T} & \leq & 1 - \alpha_{r,T} & \text{for all } r, T \end{array} \tag{S3}$$

Where $occ_{r,T}$, the average occupancy of room r during the time horizon T is given by:

$$occ_{r,T} = \frac{\sum_{t \in T} occ_{r,t}}{|T|} = \frac{\sum_{t \in T} \sum_{e \in E_t} x_{e,r}}{|T|} = \frac{\sum_{e \in E_T} x_{e,r} dur_e}{|T|} \tag{3.3}$$

Here E_T denotes the number of unique events that are scheduled during horizon T.

The occupation of a room r during slot t ($occ_{r,t}$) can be acquired using constraint H2, as at most one event can be allocated to a room during a slot t the sum of the decision variables for the given room r and slot t can only assume the values 0 (room is unoccupied) or 1 (room is occupied).

$$occ_{r,t} = \sum_{e \in E_t} x_{e,r}$$

Note that each event e can occur multiple times during a time horizon T. The number of slots (in a given horizon T) during which an event e occurs is denoted by the *duration* of event e (dur_e).

Now the proportion of rooms that have been occupied during a given time horizon T (KPI 7) can be defined by the average of the average occupation of each room during that horizon. Here the horizon T can coincide with a day, a week, the length of the scheduling horizon etc.

$$OCCn = \frac{\sum_{r \in R_n} occ_{r,T}}{|R_n|} \tag{KPI 7}$$

Where R_n denotes the set of rooms with size of at least n and $|R_n|$ the number of rooms in that set. The number of rooms that have been occupied at a given day (KPI 8) is denoted by;

$$ROOM = \sum_{r} occ_{r,T} \tag{KPI 8}$$

Where T contains all the time slots of the given day.

S4: minimise the number of different rooms per course-type.
Let E_{ctype} denote all events in course c of type *type* and $used_{ctype,r}$ a binary variable which denotes if course-type *ctype* uses room r for (at least) one of its events or not. Then $used_{ctype,r}$ takes the correct values when using the following constraints;

$$\sum_{e \in E_{ctype}} x_{e,r} \leq |E_{ctype}| used_{ctype,r} \quad \text{for all } ctype, r \tag{S4}$$

Now the number of different rooms per course-type ($rooms_{ctype}$) is calculated by;

$$rooms_{ctype} = \sum_{r} used_{ctype,r} \tag{3.4}$$

And the average number of different rooms per course-type (KPI 9) is calculated by;

$$CROOM = \frac{\sum_{ctype} rooms_{ctype}}{|ctype|} \tag{KPI 9}$$

Where $|ctype|$ denotes the number of course-types.

S5: minimise the deviation of a given initial allocation.

Let $init_{e,r}$ be a binary variable that denotes whether an event e was initially allocated to room r (1) or not (0). And let $y_{e,r}$ be a binary variable that indicates if no deviation occurs for event e and room r (0) or if it does. The deviation constraints can then be modelled by;

$$init_{e,r} - x_{e,r} \leq y_{e,r} \quad \text{for all } e, r \text{ such that } init_{e,r} = 1 \tag{S5}$$

To prevent double penalties, deviation is only penalised when an event e used to be allocated to a certain room r but will be allocated to another room, and not when an event e will be allocated to a certain room r while it was allocated to another room. Note however that when an event is allocated to multiple rooms in the initial allocation it will be penalised separately for each of these rooms. When calculating the number of modifications to a given room allocation (KPI 4) each penalised event is only counted once.

$$DEV = \sum_{e} deviated_e \tag{KPI 4}$$

Where

$$deviated_e = \begin{cases} 1, & \sum_r y_{e,r} \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

S6: minimise the space wastage.

Another way to increase utilisation is by maximising the average utilisation over all used rooms, i.e. by minimising the number of unused seats in used rooms (KPI 6). Let $sw_{e,r}$ denote the number of unused or overused seats (space wastage) for event e to room r, for the duration of event e.

$$sw_{e,r} = |cap_r - cap_e| dur_e x_{e,r} \quad \text{for all } e, r \tag{S6}$$

Where dur_e denotes the duration of event e in time slots. The total number of unused/overused seats in used rooms can be given by;

$$SPACE = \sum_{e,r} sw_{e,r} \tag{KPI 6}$$

3.1.3 Objective

The degree in which a solution satisfies the soft constraints determines the value of that solution, to find an optimal solution we need an objective function which describes how the soft constraints influence the value of a solution. In this research the objective function denotes how close a given solution is to satisfying the soft constraints, in literature this is also called a fitness function.

There are many ways to construct an objective function for an optimisation problem with multiple constraints, in this case the Lagrangian Method for Constrained Optimisation is used. The method of Lagrange Multipliers is a strategy for finding the local maxima and minima of a function subject to one or multiple equality constraints.

Let $\hat{P}$ be the optimisation problem;

$$\begin{aligned} &\text{minimise } f(x,y) \\ &\text{subject to } g(x,y) = c \end{aligned} \tag{$\hat{P}$}$$

And $\hat{\Lambda}$ the Lagrange function (or Lagrangian)

$$\hat{\Lambda}(x,y,\lambda) = f(x,y) + \lambda(g(x,y) - c) \tag{$\hat{\Lambda}$}$$

Where $\lambda(> 0)$ is called the Lagrange multiplier.

Theorem 1 (Lagrange multipliers)**.**
If f and g (as given in $\hat{P}$) have continuous first partial derivatives and if $f(x_0, y_0)$ is an optimum of $f(x, y)$ for the original constrained problem $\hat{P}$, then there exists λ_0 such that (x_0, y_0, λ_0) is a stationary point for the Lagrange function. I.e. $\nabla\hat{\Lambda}(x_0, y_0, \lambda_0) = 0$.

A stationary point is either a (local) maximum/minimum or a saddle point (a (local) maximum/minimum for on variable but not for another). Hence an minimum solution of $\hat{\Lambda}$ is a lower bound to $\hat{P}$ and possibly even a minimum solution of $\hat{P}$.

Proof ($\hat{\Lambda}$ provides a lower bound for $\hat{P}$):
Let $f(x_0, y_0)$ be an optimum of $f(x, y)$. Then

$$f(x,y) \geq min_{(x,y)} f(x,y) = f(x_0,y_0) = f(x_0,y_0) + \lambda(g(x_0,y_0) - c) \geq \hat{\Lambda}(x_0,y_0,\lambda_0)$$

Here the third step holds because the optimal solution (x_0, y_0) is also a feasible solution, hence $g(x_0, y_0) = c$ must hold. And the last step holds because $\hat{\Lambda}(x_0, y_0, \lambda_0)$ is by definition smaller or equal to all $\hat{\Lambda}(x, y, \lambda)$. □

Note that a similar theorem and proof holds for maximisation problems, then the Lagrangian provides an upper bound. This theorem and proof can be extended for multiple constraints and even inequality constraints (Karush-Kuhn-Tucker conditions).

Let $Ax = b$ the set of linear equality constraints and $Dx \leq e$ and $Fx \geq g$ sets of linear inequality constraints. Let P be the optimisation problem;

$$\begin{aligned} &\text{minimise } cx \\ &\text{subject to } Ax = b \\ &\qquad\qquad Dx \leq e \\ &\qquad\qquad Fx \geq g \\ &\qquad\qquad x \in \{0,1\} \end{aligned} \tag{P}$$

And Λ the Lagrange function (or Lagrangian)

$$\Lambda(x,\lambda) = cx + \lambda^{(1)}|Ax - b| + \lambda^{(2)}(Dx - e) - \lambda^{(3)}(Fx - g) \tag{Λ}$$

Where $\lambda = [\lambda^{(1)}, \lambda^{(2)}, \lambda^{(3)}](> 0)$ are vectors with the same dimension as respectively b, e and g. The $\lambda^{(2)}$ has a plus sign before it because $Dx > e$ should be penalised (i.e. we need to minimise $Dx - e$), the $\lambda^{(3)}$ has a minus sign before it because $Fx < g$ should be penalised (i.e. we need to maximise $Fx - g$).

Theorem 2 (Lagrange multipliers for multiple (in)equality constraints)**.**
If x_0 is an optimal solution to cx for the original constrained problem P, then there exists λ_0 such that $\Lambda(x_0, \lambda_0)$ provides a lower bound for P.

Proof.
Let x_0 be an optimal solution to P. Then

$$\begin{aligned} cx &\geq min_x\ cx = cx_0 \\ &\geq cx_0 + \lambda^{(1)}|Ax_0 - b| + \lambda^{(2)}(Dx_0 - e) - \lambda^{(3)}(Fx_0 - g) \\ &= \Lambda(x_0, \lambda) \\ &\geq \Lambda(x_0, \lambda_0) \end{aligned}$$

Here the second step holds because $Ax = b$, $Dx \leq e$ and $Fx \geq g$ for all x and hence also for x_0 and because $\lambda > 0$. The last step holds by definition of $\Lambda(x_0, \lambda_0)$ (the optimal value for Λ). Hence $\Lambda(x_0, \lambda_0)$ provides a lower bound for P. □

Kerrigan and Maciejowski [2000] show how to use the method of Lagrange to optimise an optimisation problem with soft inequality constraints. Instead of adding the hard constraints to the Lagrangian they add the penalty variables to both the optimisation problem and the Lagrangian.

$$\begin{aligned} &\text{minimise } cx \\ &\text{subject to } |Ax - b| = \epsilon^{(1)}(x) \\ &\qquad\qquad Dx \leq e + \epsilon^{(2)}(x) \\ &\qquad\qquad Fx \geq g - \epsilon^{(3)}(x) \\ &\qquad\qquad x \in \{0,1\} \\ &\qquad\qquad \epsilon^{(1)}(x), \epsilon^{(2)}(x), \epsilon^{(3)}(x) \geq 0 \end{aligned} \tag{P_ϵ}$$

And Λ_ϵ the Lagrange function (or Lagrangian)

$$\Lambda_\epsilon(x, \epsilon(x)) = cx + p^{(1)}\epsilon^{(1)}(x) + p^{(2)}\epsilon^{(2)}(x) + p^{(3)}\epsilon^{(3)}(x) \tag{Λ_ϵ}$$

Where $p^{(i)}$ is called the penalty weight. They show that is p is chosen such that $p \geq f(\lambda)$ (for more information on $f(\lambda)$ see their paper) then the optimal solution to Λ_ϵ is equal to the optimal solution of P (if any exists). If P has no optimal and thus feasible solution then we can use Theorem 2 to show that Λ_ϵ provides a lower bound for P_ϵ (with $p^{(i)} = \lambda^{(i)}$).

The objective function can now be described as followed;

$$
\begin{aligned}
minimise \sum_{e,req} & y_{e,req} pen_{e,req} && \text{(from constraint S1)} \\
+ \sum_{e,r} & z_{e,r} pen_{e,r} && \text{(from constraint S2)} \\
+ \sum_{r,T} & z_{r,T} pen_{r,T} && \text{(from constraint S3)} \\
+ \sum_{ctype} & (rooms_{type} - 1) pen_{ctype} && \text{(from constraint S4)} \\
+ \sum_{e,r} & y_{e,r} pen_{e,r} && \text{(from constraint S5)} \\
+ \sum_{e,r} & sw_{e,r} pen_{e,r} && \text{(from constraint S6)}
\end{aligned}
$$

In this research we assume that the penalty weights are given by the stakeholders, hence $p^{(i)}$ can not be chosen (or optimised). The penalty variables are denoted in gray. Note that a course type needs at least one room, $rooms_{ctype}$ should therefore be penalised if it takes on values larger then 1, hence $rooms_{ctype} - 1$ is used in the objective function.

3.2 Reducing complexity

This section proposes two ways to reduce the complexity of the CAP ILP model as given in Section 3.1. There are numerous other ways to reduce complexity, for example by decomposing the problem in time (days, weeks, etc.). As these are not used in this research they will not be described here.

3.2.1 Decomposition in room size

This technique makes use of constraint H3, i.e. that in a feasible solution the capacity of a room should be at least as large as the capacity of the event you allocate to it. This means that a set of events which have a capacity larger or equal to s can only be allocated to rooms which have at least capacity s. Hence events and rooms can be partitioned into G groups according to their capacity, i.e. they are decomposed by size.

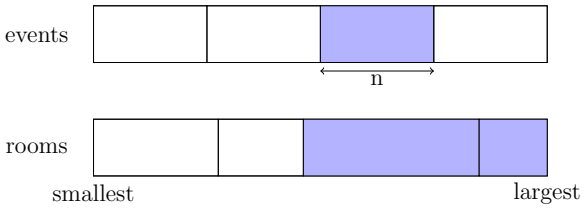


Figure 3.1: Events and rooms for the second sub problem

Let each subproblem $i \in \{1, ..., G\}$ consist of approximately n events and let e_i be the smallest event in group i. Then the events in subproblem i need to be allocated to rooms which have a capacity of at least cap_{e_i}, i.e. all rooms with a capacity of at least cap_{e_i} belong to subproblem i. Hence each sub problem will have around n events, but the number of rooms will increase for each sub problem, until all rooms are needed (see example). With each subproblem that is solved the availability of the rooms decreases, as they are allocated to events or a conflict-room (see hard constraint H4) has been allocated to an event.
The events are allocated by decreasing size to prevent large events not being allocated. Note that events of the same size are always put in the same group, this might lead to groups of different sizes.

The technique works as followed;

- Sort events and rooms by size
- Determine average group size $n = \lceil \frac{|E|}{G} \rceil$
- Check the size of events $|E| - n + 1, |E| - 2n + 1, ..., |E| - (G-1)n + 1$
- Put all events with size of $cap_{e_{|E|-n+1}}$ or more in group 1
- Put all events with $cap_e \in [cap_{e_{|E|-2n+1}}, cap_{e_{|E|-n+1}})$ in group 2
- ...
- Put all events with size less then $cap_{e_{|E|-(G-1)n+1}}$ in group G
- Put all rooms with size of $cap_{e_{|E|-i \cdot n+1}}$ or more in group i

The technique will be explained using the example given in Tables 3.1 and 3.2. The example problem will be partitioned in three groups ($G = 3$).

Table 3.1: Size decomposition example: Events

Name	e1	e2	e3	e4	e5	e6	e7	e8	e9
Size	10	20	30	30	30	50	60	60	70
Time slots	1,2	2	1	1	2	2	1,2	1	2

Table 3.2: Size decomposition example: Rooms

Name	r1	r2	r3	r4	r5	r6
Size	10	30	40	50	60	70

Example:
As the example problem needs to be partitioned in three groups, each group has on average three events. We therefore check the size of events $e4$ and $e7$, which correspond to 30 and 60. As event $e3$ also has size 30 it will be placed in the same group as event $e4$. The groups (subproblems) are now given by;

- group 1: Events e7, e8 and e9. Rooms r5 and r6.
- group 2: Events e3, e4, e5 and e6. Rooms r2, r3, r4, r5 and r6.
- group 3: Events e1 and e2. All rooms

When events are put in the best fitting room (minimise seat under-use, don't allow seat overuse) this leads to the schedule given in Figure 3.2. In this small example such a result could easily be retrieved by hand, when the number of events and rooms increase the complexity increases as well. Table 4.3 shows that decomposition in size can reduce the number of needed decision variables by over 50%.

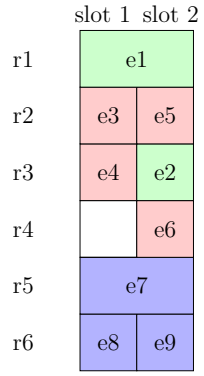


Figure 3.2: Size decomposition example: Allocation

3.2.2 Decomposition for dynamic allocation

When the allocation is already operational, measurements can indicate bottlenecks; events that do not fit in their allocated room any more or leave more seats unused than preferred. In those cases it can be decided that similar events (events of the same course-type) should be reallocated using new estimated capacities. Completely reallocating all events is complex and not preferable, some events should not be reallocated (unless they become a bottleneck event) while others are still flexible. Events which have their start time in the past are fixed by default. Other examples of fixed events can be (large) lectures, reservations and practicals.

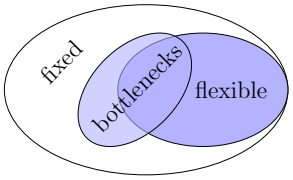


Figure 3.3: Events and rooms for the second sub problem

Reallocating bottlenecks and flexible events each time a bottleneck is detected is not preferable, it can give too many modifications for flexible events and does not help the detected bottleneck anyway. Detections of bottleneck events are therefore collected and reallocated at the end of a predefined horizon, e.g. a day or a week. At such a point in time all flexible events and bottleneck events are collected and reallocated to available rooms. Note that room-slots which were allocated to flexible events are available in this problem, while room-slots allocated to fixed events are not available.

Chapter 4

Setup and Results: Automatic Room Allocation

This chapter describes the results when automatically producing a room allocation using the mathematical model in Chapter 3. Section 4.1 gives an overview of the data used in this research and chosen values for various parameters. Section 4.2 illustrates the results of the mathematical model and compares results for different objectives.

4.1 Setup

The room allocations made in this research are based on, and compared to, the actual timetable of the first Quartile of the year 2014-2015 at the UT. This period was first off all chosen because it had the most recent data available for a complete term. The year 2014-2015 is also the first year that the pilot studies have all of their (new) Bachelor students use the new educational system. It is therefore interesting to see what the room allocation for this new system looks like and on which points rooms or education could be improved to suit each other better.

The first Quartile of a year is also interesting for this research because student estimation and real student enrolment differ the most at the start of the year, due to the large intake of students in September. It has been hypothesised that the first Quartile is also the most dynamic in terms of student attendance; students are still adapting to their new study (year) and are often more motivated to follow lectures at the start of the year. Or students find out that their chosen study was not what they expected from it, in which case they often change or stop their studies. The deviation between student estimation and student enrolment or attendance can lead to changes in room allocation before term and during term.

Quartile 1A (Q1A) starts at September 1^{st} and ends on November 7^{th}. No education is planned in weekends and reservations booked by students (via Web Room Booking[1]) can only occur during weekdays. Weekends are therefore omitted from this research, which causes Q1A to consist of 10 weeks of 5 days each.
Most education is planned during the day, during the nine blocks of 45 minutes described in Table 4.1. Some events however span multiple blocks (including the intermediate breaks) or start earlier or later, the scheduling team therefore works with planning blocks (time slots) of 15 minutes (the greatest common divisor).

Table 4.1: Block hours

block	starttime - endtime
1	08:45 - 09:30
2	09:45 - 10:30
3	10:45 - 11:30
4	11:45 - 12:30
5	12:45 - 13:30
6	13:45 - 14:30
7	14:45 - 15:30
8	15:45 - 16:30
9	16:45 - 17:30

Because almost no education is planned during the evening, evening hours will not be taken into account in this chapter. This leads to a planning day from 8:45 up to 17:30, which corresponds to 35 time slots per day. At the start of a Quartile no reservations are made yet and only a small amount of the educational events need small (project) rooms, of which there are plenty. No sophisticated algorithm is needed for these rooms and events, and these largely unoccupied rooms decrease the average overall occupation. The small project rooms and events are therefore omitted from this set-up. They are needed however when reservations are taken into account.

Other data that is omitted from this set-up are; events that do not need a room (at the UT), events and rooms at the ITC[2] and reservations for maintenance, promotion, meetings etc.

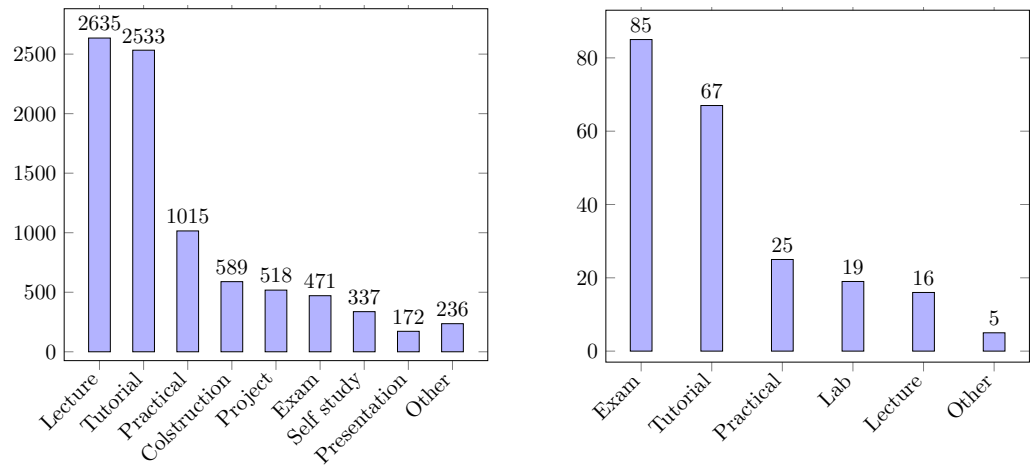
The characteristics of Quartile 1A are summarised in Table 4.2.

Table 4.2: Characteristics of Quartile 1A, from 09-09-14 up to 07-11-14

start time	end time	events	rooms	properties	time slots	programmes	courses	course-types
08:45	17:30	8506	217 (128)	38	1750	195	509	775

[1]Online reservation tool for project rooms
[2]The Faculty of Geo-Information Science and Earth Observation is scheduled separately

The first Quartile consists of 8506 educational events which can be part of one of the 195 different programmes and 509 courses. Note that programmes and courses (as defined in the model of this report) do not have a one to one correspondence with curricula, modules, courses, module sections and course sections. This differs per faculty and/or study. Each event has its own event-type, an overview of the number of events per type is given in Figure 4.1a. Here a colstruction is a relatively new event-type; a combination of a lecture and a tutorial, which needs a room that can provide both.
Note that Figure 4.1 shows the events which occur (only once) during Quartile 1A and the rooms (which can be used multiple times per day) which are available in Quartile 1A.



(a) Number of events per type

(b) Number of rooms per type

Figure 4.1: Distribution of event- and room-types

The UT virtually has 217 rooms at its disposal for educational events (when excluding the project rooms), in reality this number is lower (128). Some rooms can be combined or split or used for exams (in which case not all chairs are allowed to be occupied[3]). Constraint H4 ensures that a combined room and one of its partial rooms cannot be used together, and that a room and its exam version are never used at the same time.

All virtual rooms have their own room-type, as can be seen in Figure 4.1b. Most of the rooms are tutorial rooms or (versions of) rooms which can be used for exams. Each room can satisfy, and each event can require, more than one of the 38 existing requirements. Satisfying or requiring zero requirements is allowed, but not common. The requirements refer to either room-type, equipment, size or a combination of those.

[3]Space should be created around exam participants to discourage cheating.

To compute an optimal room allocation for 8506 events and 217 rooms, over 1.8 million decision variables are needed. To decrease the complexity of this problem the size decomposition technique of Section 3.2.1 is used. In Table 4.3 an overview is given of the number of decision variables needed when decomposing the problem into multiple sub problems.

Here the bounds denote size of the smallest event (lower bound) of each sub problem, the lower bound of zero is omitted from the table. The second and third column denote the number of events and rooms per sub problem, events are not always equally distributed over each sub problem as events of the same size are always assigned to the same group.

Table 4.3: Size complexity when decomposing in size

# sub-problems	bounds	# events	# rooms	# decision variables	reduction
1	-	8506	217	1.85E6	-
2	36	4409, 4097	92, 217	1.29E6	29.9%
3	52, 30	2860, 2869, 2777	59, 112, 217	1.09E6	40.8%
4	**70, 36, 24**	**2402, 2007, 2026, 2071**	**46, 92, 143, 217**	**1.03E6**	**44.0%**
5	80, 45, 30, 20,	1765, 1804, 2160, 1233, 1544	39, 77, 112, 170, 217	9.94E5	46.1%
6	82, 52, 36, 30, 18	1418, 1442, 1549, 1320, 1405, 1372	35, 59, 92, 112, 171, 217	9.63E5	47.8%
7	90, 64, 40, 35, 25, 15,	1267, 1176, 1512, 934, 1279, 1297, 1041	32, 48, 92, 98, 128, 183, 217	9.55E5	48.3%

It can be seen that for this particular instance decomposition in two sub problems already reduces the number of decision variables by more than 25%. Note that when the number of sub problems increases the total number of decision variables decreases, the overall processing time (splitting, solving and merging the sub problems) increases however. For the experiments in Section 4.2 decomposition in four sub problems is chosen.

4.2 Results

This section gives an overview of some of the results obtained from the mathematical model in Chapter 3. The intention of this section is to show how the soft constraints (wishes in Section 2.3) influence the measured KPI values. This is done by answering the following questions;

1. Can we find a feasible room allocation for the given set-up (see Section 4.1)? And if not, why not?
2. Can we approximate the actual room allocation in Quartile 1A? What are the differences between the actual room allocation and the automatically produced room allocation?
3. How does room suitability influence the allocation and what happens when rooms become more flexible?
4. Can we obtain an average occupation of 70%? And what does this do to the other KPI measures? Is this an acceptable lower bound?
5. What average utilisation can we obtain? And what does this do to the other KPI measures?
6. Can we decrease the amount of rooms per course(type)?

These questions originate from meetings with supervisors, management and literature.

Before looking at the influence of wishes to the room allocation it is sensible to look if there is a feasible solution to the CAP problem at all. Also interesting is to look if the actual room allocation, made by the scheduler team, can be approximated. The difference (deviation) between these allocations can point out benefits and shortcomings of the automatic allocation.

Question 3 determines how room suitability influences the room allocation and if more flexible rooms are needed at the UT. Question 4 and 5 show the influence of occupation (soft constraint S3) and utilisation (soft constraint S2) on the room allocation. While Question 6 looks at the influence of constraint S4 (minimise rooms per course-type) on the number of different rooms per course(type) and the other KPIs.

4.2.1 Feasibility

This subsection investigates the feasibility of the given set-up, i.e. can we find a feasible room allocation? If not, it will show why such a feasible allocation is not possible. In this case only the hard constraints (as given in Section 3.1.1) are used. Table 4.4 shows the feasibility KPIs; Allocation (KPI 1) and Misfits (KPI 2). It can be seen that all events can be allocated but that three events do not fit in their assigned room. These three events are large mathematics lectures which each need 9 more seats than the largest (non-exam)room can provide.

In reality the schedulers will ask the corresponding teacher if the lecture should be split up into smaller groups or if the current room is sufficient enough (read: if the teacher does not mind the overuse of seats or the probability of all students attending is less than 1). In this report the number of students for these specific lectures are reduced by 9, this way the rest of the results will always provide a feasible solution.

Table 4.4: Results when using hard constraints only

week	#Events	Allocation	Misfits
36	960	100%	1
37	997	100%	1
38	1090	100%	1
39	974	100%	0
40	952	100%	0
41	998	100%	0
42	941	100%	0
43	820	100%	0
44	446	100%	0
45	328	100%	0
0	8506	100%	3

4.2.2 Deviation

This subsection examines how close the automatic allocation can approximate the actual allocation during Quartile 1A. This deviation (D) is minimised using soft constraint S5. The KPIs most important to the schedulers are; room requirements (KPI 3,R) and seat under-/overuse (KPI 6,S). Soft constraints S1 and S6 are therefore also added to the objective. To show the influence of each of the constraints we look at the following cases;

1. Only look at deviation: use soft constraint S5 (D)
2. Take deviation, room requirements and space under-/overuse into account. Use soft constraints S5, S1 and S6 (DRS)
3. Take room requirements and space under-/overuse into account. Use soft constraints S1 and S6 (RS)

The first case is to see if the initial allocation provided by the scheduler team satisfies the hard constraints set in this report. When such a hard constraint cannot be met the model will deviate from the initial allocation. The second case is added to fine tune these deviations to the initial allocation, i.e. to tell the model what sort of room it should choose when the room it was allocated to initially (i.e. by the scheduler team) is not available (due to hard constraints set in this report).
The third case is to see if the model can produce an acceptable allocation (in terms of room requirements and space under-/overuse) when an initial allocation is not provided, i.e. can the model produce an acceptable allocation from scratch?

Figure 4.2 shows that the first case (D) can closely approximate the actual allocation during Quartile 1A, except during the last two weeks. These weeks hold a lot of exams which often share big exam rooms. As the model in this report cannot put exams (or events of other types) together in one room, the deviation is very large in these weeks.
The deviation in the rest of the weeks is largely due to events which were given no size and are normally scheduled by hand. Deviation could also occur because because an event has requirements which are not recorded in Syllabus+ (and therefore not in this research). I.e. there might be rules to the allocation process which are not known or recorded. These unwritten rules may sound obvious to a person but not to a computer.

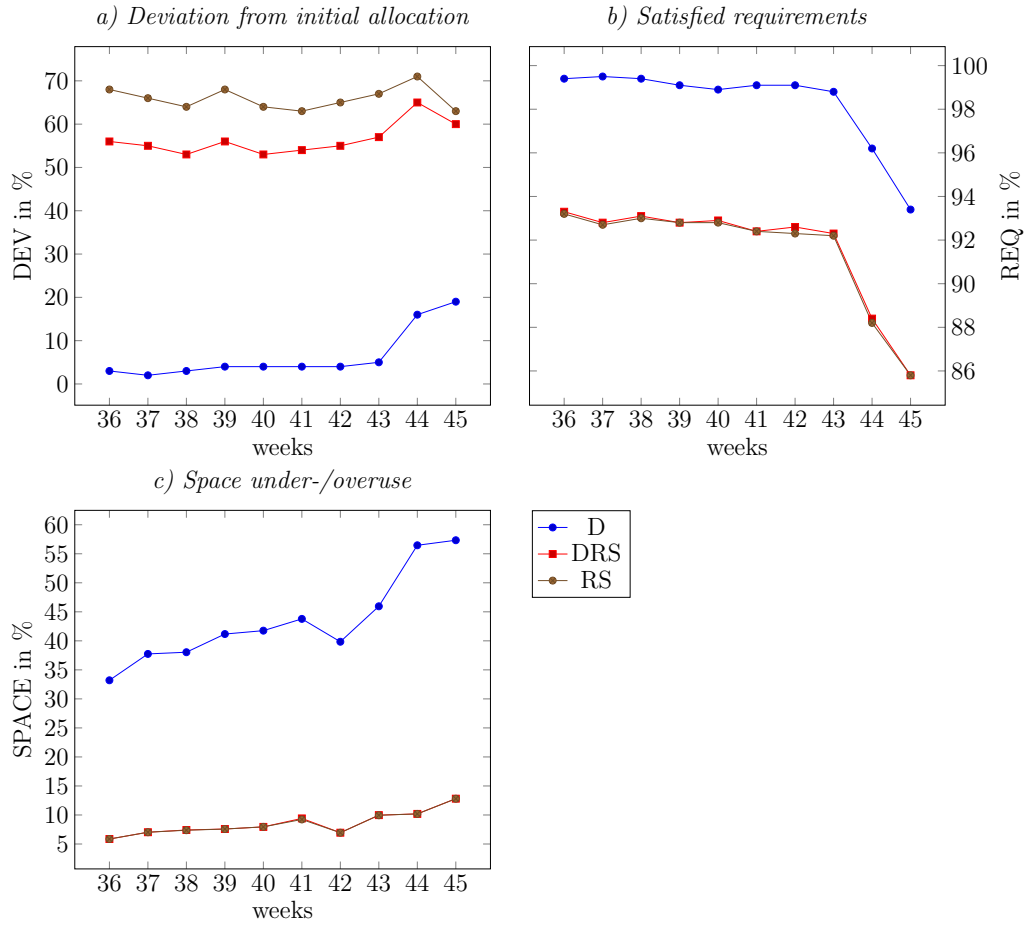


Figure 4.2: KPI measurements for Deviation (D), Deviations & Requirements & Space (DRS) and Requirements & Space (RS)

In the first 8 weeks, when the deviation case (D) approximates the actual allocation closely the percentage of satisfied requirements is very high. We can therefore conclude that the scheduler team performs very good on this measure (during these weeks). An improvement can however be made in terms of space under-use (see Figure 4.2c, case DRS and RS). However, to decrease the space under-use one has to deviate (see Figure 4.2a, case DRS and RS) from the actual allocation and sacrifice some of the, by the deviation case (D), satisfied requirements.

This leads to the case in which also requirements and space under-/overuse are taken into account (DRS). This case performs much better on the SPACE measurement, but worse on the REQ measurement (see Figure 4.2b). In the DRS case all three objectives have equal weight/priority, if space under-/overuse was given less weight a higher space under-/overuse (but not as high as for case D) and a higher satisfaction of requirements (but again not as high as for case D) might be found.

The third case (RS) deviates even more from the actual allocation and performs approximately the same as the second case (DRS). In some weeks the RS case performs a bit worse (on requirements) than the DRS case. The DRS case (and the D case) can however make use of the actual allocation, which already performs very good on satisfying requirements. While the RS case does not use the data of the actual allocation at all, it is a completely automatic generated allocation.

From this we can conclude that an automatically generated allocation can outperform the actual allocation on space under-/overuse and give a reasonable value on room requirements (almost as good as the DRS case).
This depends however on KPIs that are taken into account and which priority is given to them. Changing the weights of the different soft constraints can give better performance on one KPI (SPACE in this case) but worse on another (REQ in this case).

Another conclusion, that was made in a previous paragraph, is that the last two weeks of Quartile 1A are difficult the approximate due to events sharing rooms. It is therefore also hard to draw any conclusions on the KPI measures during these weeks (as they do not reflect reality well). These two weeks (week 44 and 45) are therefore omitted in the rest of this chapter.

4.2.3 Room suitability

This subsection looks at the influence of room requirements on the allocation. It shows why events are sometimes allocated to a bigger room than (seemingly) needed. This section also looks at the case in which rooms are more flexible, i.e. what happens to the allocation when room requirements do not need to be taken into account. The corresponding KPIs for this question are space under-/overuse (KPI 6,S), room requirements (KPI 3,R), utilisation (KPI 5,U) and occupation (KPI 7,O). To simulate a case in which all rooms are flexible, the room requirement constraint is omitted. The base case will take both space under-/overuse and room requirements equally into account and a third case will prioritise room requirement over space under-/overuse.

1. Take room requirements and space under-/overuse equally into account: Use soft constraints S1 and S6 and give each a weight of 1 (SR)

2. Take only space under-/overuse into account. Use soft constraint S6 (S)

3. Take room requirements and space under-/overuse into account: Use soft constraints S1 and S6, but give room requirement a weight of 10 and space under-/overuse weight of 1 (SR10)

Before looking at the usage of rooms and seats, we will first look at the values of deviation and satisfied room requirements. Figure 4.3 gives an overview of the KPIs deviation (KPI 4) and room requirements (KPI 3). For clarity the weight of the room requirements KPI is denoted in each graph. Requirements are either not taken into account (0, S), equally taken into account (1, SR) or given priority (10, SR10).

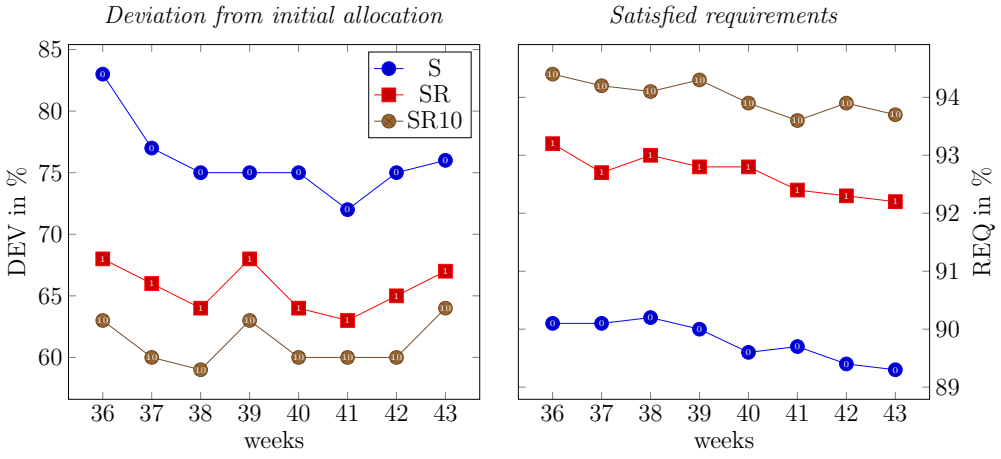


Figure 4.3: Deviation and satisfied requirements for Space (S), Space & Requirements (SR) and Space & Requirements (extra important) (SR10)

The rightmost graph shows us that giving a higher weight to the room requirement KPI gives, as expected, a better result on that KPI. Although even when no weight is given to room requirements most of them are still satisfied, hence the majority of the rooms seem to satisfy the most required requirements.

The leftmost graph shows us that when the weight on room requirement is set higher, the deviation from the actual allocation decreases. This implies that the room requirement objective was given high priority when creating the actual room allocation.

Figure 4.4 shows results of the utilisation (KPI 5) and space under-/overuse (KPI 6) KPIs. When room requirement is given higher priority the results of utilisation and space under-/overuse decrease. Hence there is indeed a trade off between room requirements and room sizes. The difference in utilisation and space under-/overuse between S (ignoring room requirements) and SR (giving room requirement and space under-/overuse equal priority) is however small.

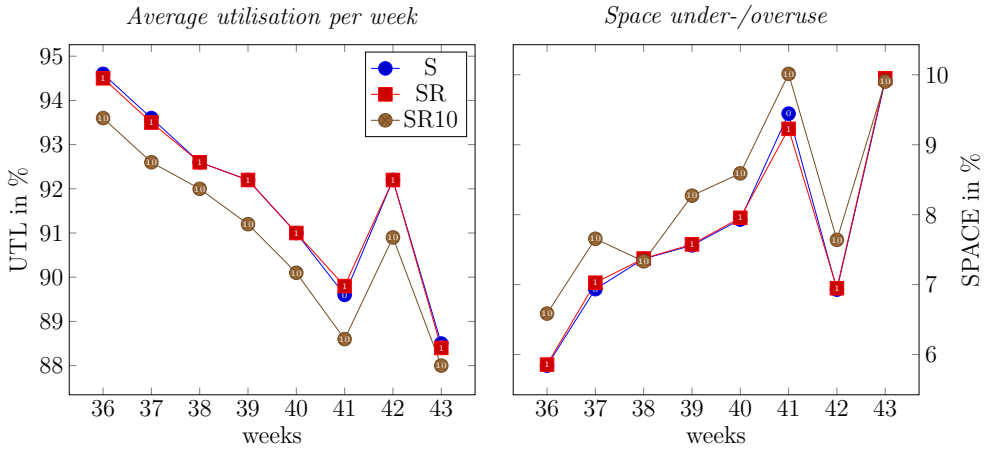


Figure 4.4: Utilisation and space under-/overuse for Space (S), Space & Requirements (SR) and Space & Requirements (extra important) (SR10)

The rightmost graph in Figure 4.4 even shows that in some cases (week 38) this difference is insignificant even when having a weight of 10 for the room requirements. Hence it is possible that an event is allocated to a bigger room than (seemingly) needed such that the requirements can be satisfied, but most of the times a fitting room can be found (as the average utilisation never drops below 88%).

4.2.4 Occupation

This subsection shows the influence of setting a lower bound on the occupation of rooms. In an interview with the facility service center at the UT Dijksterhuis asked about their required norm for room occupation (Dijksterhuis [2014]). They have set the norm at 70%, but use it mainly as an indicator for when to build extra rooms.

The corresponding KPI for this question is, obviously, occupation (KPI 7). As all events should be allocated to exactly one room, and the number of events do not change between cases, neither will the number of used rooms. And hence the overall average occupation will always be the same for each feasible solution. Therefore we will look at the distribution of room-occupation per day using a box-plot. A box-plot shows several statistics on data; the minimum value, the maximum value, median (the most often occurring data), IQR (the middle 50% of the data) and outliers.
To show the influence of the occupation constraint (soft constraint S3) we will compare four cases. Three of the four cases will take the room requirement kpi (KPI 3) into account, as it could be concluded from the previous subsections that this KPI is very important at the UT. The last case does not and is used to see if room requirements have influence on the occupation. In all cases the occupation is calculated per day.

1. Do not set a lower bound on the occupation (aka a lower bound of 0): Use soft constraint S1 (R)
2. Set a lower bound of 50% on the occupation of a room: Use soft constraints S1 and S3 (RO50)
3. Set a lower bound of 70% on the occupation of a room: Use soft constraints S1 and S3 (RO70)
4. Set a lower bound of 50% on the occupation of a room do not take room requirements into account: Use soft constraints S3 (O50)

Figure 4.5 shows the box-plots for all four cases. In all cases empty rooms (rooms that are empty for the complete day) are ignored, as rooms are not penalised when they are empty.
A quick glance can tell that the differences between the four cases are minor. So either there is not much leeway in the given data or there is a bug in the constraints. We can at least conclude that the room requirement constraints do not lead to (much) less leeway for the occupation constraint, as case O50 produces about the same results and does not take room requirements into account. There are however cases in which room requirements do restrict the occupation outcome, e.g. on day 11 75% of the rooms in case O50 have an average occupation of at least 30%, while in case RO50 this is more in the direction of 60%.

If we compare case RO70 with the cases with lower lower bounds (R and RO50) the median (red line) and middle 50% of the data (blue blocks) do seem to be a bit higher than the other cases. But on some days it seems to perform worse, e.g. on day 5 the blue block start lower for case RO70 than case R. The median however is higher for case RO70 than for case R.

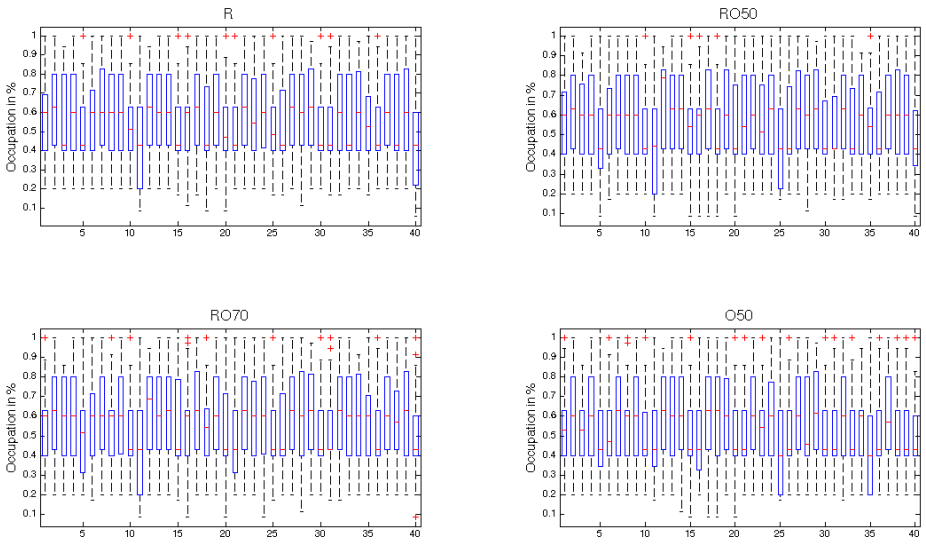


Figure 4.5: Distribution of occupation per day for Requirements (R), Requirements & Occupation for various lower bounds (RO50,RO70) and Occupation with a lower bound of 50% *(O50)*

When we look at the aggregated penalties (the sum of the $z_{r,t}$ values for all rooms r and all days t) we get even stranger results. Figure 4.6 shows the aggregated occupation penalty for several lower bounds. It shows that the O50 case, the only case without room requirements, is penalised more than those with room requirement constraints. Although this could be explained by the fact that if events require the same requirements they might end up in the same room (which satisfies the combination of these required requirements).
More research is however needed to show if this hypotheses is true and why the occupation constraints do not behave as expected.

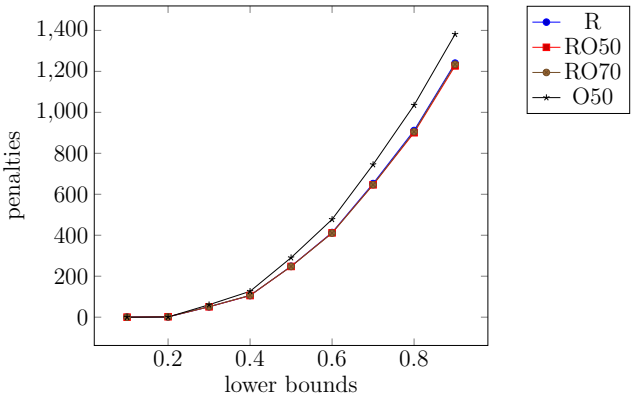


Figure 4.6: Aggregated penalties for several lower bounds for the cases Requirements (R), Requirements & Occupation for various lower bounds (RO50,RO70) and Occupation with a lower bound of 50% *(O50)*

4.2.5 Utilisation

This subsection shows the influence of soft constraints S2 (minimise utilisation) and S6 (minimise space under-/overuse) on the utilisation (KPI 5) and other KPI measures. We will look at how good the utilisation can be steered with the use of these constraints. In this subsection the following cases will be compared, note that all cases use the requirement constraint.

1. Set a lower bound of 50% on the weekly average utilisation of a room: Use soft constraints S1 and S2 (RU50)
2. Set a lower bound of 70% on the weekly average utilisation of a room: Use soft constraints S1 and S2 (RU70)
3. Set a lower bound of 90% on the weekly average utilisation of a room: Use soft constraints S1 and S2 (RU90)
4. Take space under-/overuse into account: Use soft constraints S1 and S6 (RS)
5. Take space under-/overuse into account, but with lower priority: Use soft constraints S1 and S6, but give room requirements a weight of 1 and space a weight of 0.1 (RS.1)

As the space constraint has big impact on the SPACE and REQ kpis (see Figure 4.2) a lower priority (0.1) is given to SPACE in the last case. This way we can research what effect tuning the penalty parameters has on the various kpis.

Figure 4.7 shows the measurements for the KPIs deviation (KPI 4), room requirements (KPI 3) and average utilisation per week (KPI 5). The deviation from the actual allocation in Quartile 1A is biggest for case RS (just as we saw in Figure 4.2). This is caused by the SPACE constraints which makes this case deviate more from the actual allocation, have a decreased performance on room requirements (but not really bad) and a much better performance on both SPACE and average utilisation (see Figure 4.2c and Figure 4.7c).

When comparing cases RU50, RU70 and RU90 for the percentage of satisfied room requirement (REQ) we see that U50 outperforms RU70 which outperforms RU90, hence if one desires a higher average utilisation one might have to dissatisfy more of the room requirements. A higher lowerbound on the utilisation does however not imply a better utilisation according to Figure 4.7c, as RU50 outperforms RU90, i.e. has a higher utilisation. And this happens even though an average utilisation of 90% should be possible (see case RS).

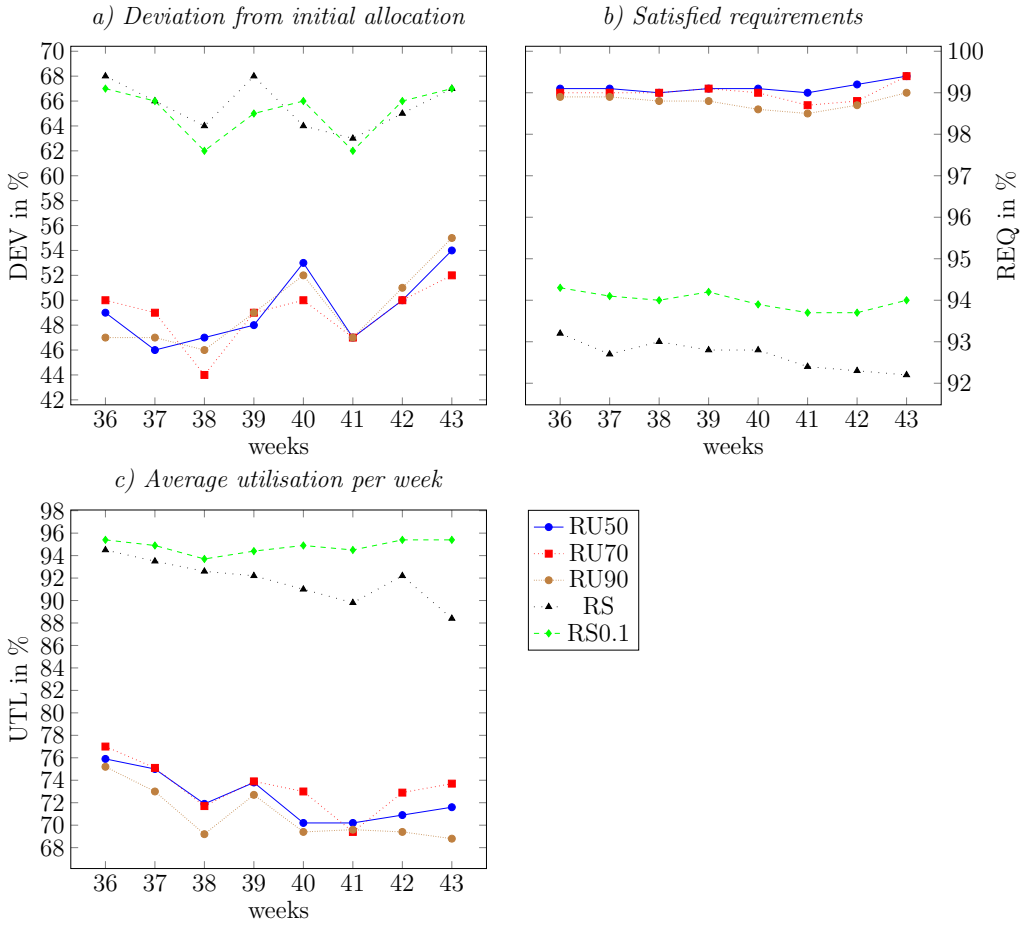


Figure 4.7: KPI measurements for the cases Requirements & Utilisation with various lower-bounds (RU50,RU70,RU90), and Requirements & Space under-/overuse with various priorities (RS, RS0.1)

If we compare the cases which use the utilisation constraint (RU50,RU70,RU90) with the cases which use the space under-/overuse constraint (RS,RS0.1) we can see that the latter two perform much better on the UTL kpi, although this leads to less satisfied requirements and more deviation from the initial allocation. This is good news, as the SPACE constraints are easier to set-up and faster to solve.

If we compare both cases which use the SPACE constraint we see that lower priority on SPACE, and hence more on room requirements, indeed leads to a better satisfaction of room requirements. Strangely it also leads to a better utilisation (and also less space under-/overuse). It can be concluded that more research should be done on the influence of priority (penalty parameters) on the various KPIs.

4.2.6 Course room stability

This subsection investigates the course room stability constraint (soft constraint S4). An overview is given of the average number of different rooms per course(type), the number of course types which need more than one room and the maximal extra rooms needed per course. The influence on the KPI measures deviation (KPI 4), room requirements (KPI 3) and space under-/overuse (KPI 6) is also shown.

In this subsection the three cases are compared, all three cases will take the room requirement kpi (KPI 3) and space under-/overuse kpi (KPI 6) into account.

1. Take only room requirements and space under-/overuse into account. Use soft constraints S1 and S6 and give each a weight of 1 (SR)

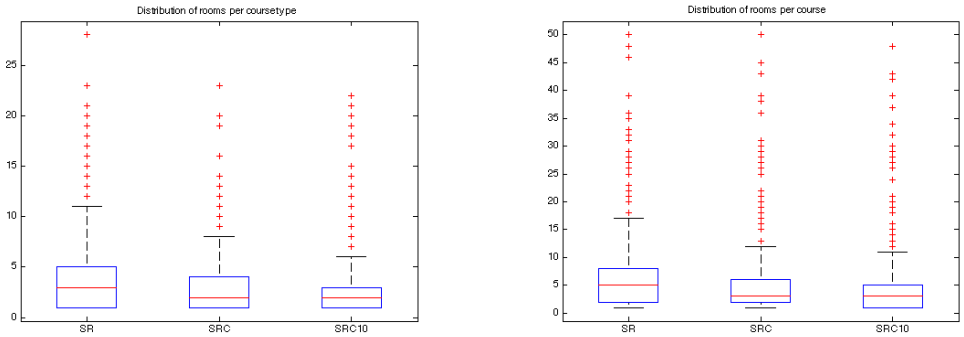
2. Take room requirements, space under-/overuse and course room stability equally into account: Use soft constraints S1, S6 and S4 and give each a weight of 1 (SRC)

3. Take room requirements, space under-/overuse and course room stability into account: Use soft constraints S1, S6 and S4, but give course room stability a weight of 10 and the others a weight of 1 (SRC10)

Quartile 1A has 775 different course-types, Table 4.5 shows statistics on room assignment per course(type). The CROOM constraint reduces the number of different rooms to one for another 41 course-types and when given priority (weight 10) for another 73 course-types.

Table 4.5: Statistics on number of rooms per course(-type)

	course-types which need only 1 room		average number of rooms		median number of rooms	
case	amount	percentage	per course-type	per course	per course-type	per course
SR	221	28.5%	3.82	6.84	3	5
SRC	262	33.8%	3.05	5.47	2	3
SRC10	335	43.2%	2.66	4.76	2	3



(a) Distribution of rooms per course *(b) Distribution of rooms per course-type*

Figure 4.8: Distribution rooms per course(type)

On average case SR used 3.82 rooms per course-type and 6.84 rooms per course, this can be decreased to 2.66 and 4.76 respectively using soft constraint S4 with a weight of 10. The most common number of rooms per course(type) (the median) are lower however, because there are some outliers (see Figure 4.8). These outliers are mostly course-types that have several student-groups which need their own (project)room at the same time, and hence a lot of different rooms.

Figure 4.8 shows that the more priority is given to the CROOM constraint the more the number of rooms per course(type) decreases. This has however consequences for the deviation and percentage of satisfied requirements, the deviation becomes larger when more priority is given to CROOM (see Figure 4.9a) and the percentage of satisfied requirements drops (see Figure 4.9b). Interestingly the space under-/overuse improves when CROOM is taken into account (see Figure 4.9c). It could be that the CROOM constraint chooses a room which has a low space under-use, while case SR prefers a room with more seats but which satisfied more requirements.

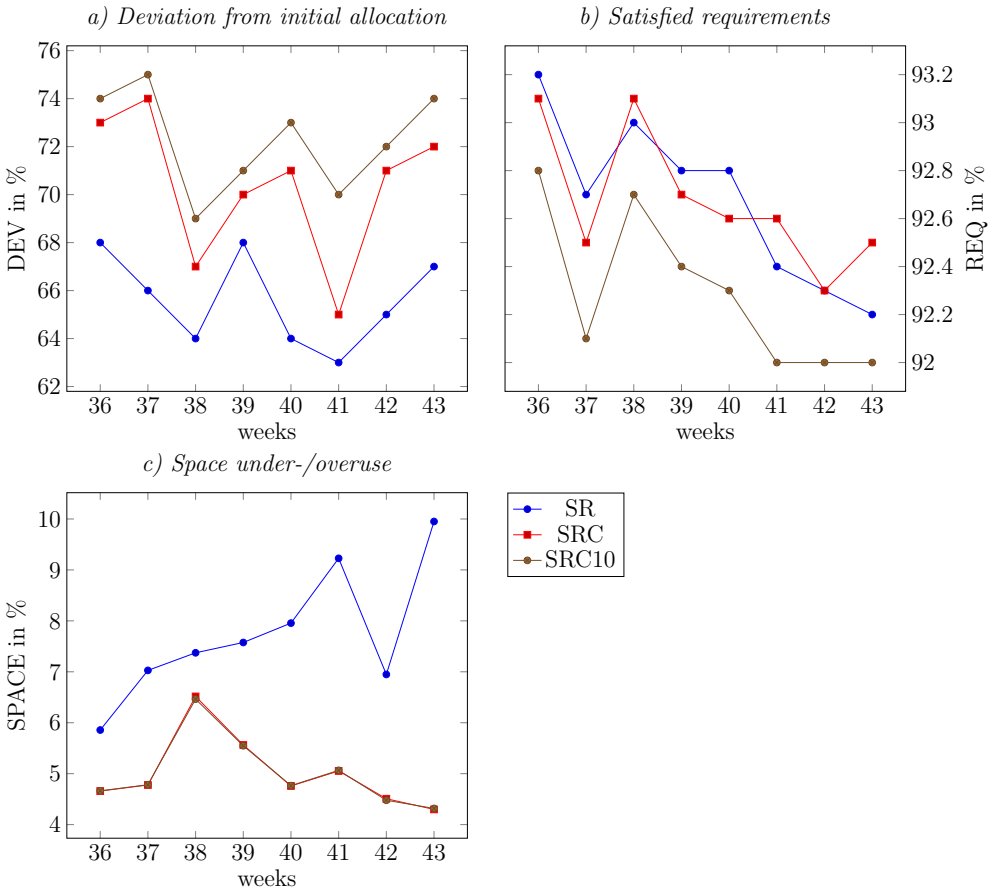


Figure 4.9: KPI measurements for the cases Space under-/overuse & Requirements (SR), Space under-/overuse & Requirements & Course room stability with various priorities (SRC,SRC10)

Dynamic allocation

Chapter 5

Simulate a Quartile

This chapter describes the approach to simulate a Quartile. Section 5.1 tells how course enrolment, and specifically changes to the expected course enrolment, is simulated. The topic of Discrete Event Simulation is introduced in Section 5.2. This technique is used to simulate the evolvement of the room allocation as it adapts to data from scanners and incoming reservations. In Section 5.3 it is shown how new events (e.g. reservations) are simulated and Section 5.4 gives an overview of the algorithm used to allocate reservations to a room.

5.1 Simulate course enrolment and attendance

As soon as the real course enrolment data is known modifications should be made for courses which have more or much less students than scheduled for. Data on course enrolment can be retrieved from the student information system OSIRIS. The distribution of students over different tutorial groups, project groups and such differs however per course/module. It would introduce a lot of programming logic and data to determine the new number of students per event for a certain course. Hence another, less realistic, method is used to simulate the course enrolment. For each course c;

1. Course c is changed with probability $Pchanged$.
2. All events in course c increase/decrease with percentage $percentage\_change$, this percentage is chosen from a predefined set (each element i in this set can be chosen with probability $Pchange_i$). To prevent non-integer sizes and empty events all new sizes are rounded up. I.e.

$$\forall_{e \in c} \quad new\_size = \lceil percentage\_change \times old\_size \rceil$$

The same method (only step 2) is used in the next section to estimate the new future attendance (size of all future events) in a course when the attendance (size) of the detected event (in that same course) has changed. A bit of randomness can be added by defining a noise parameter η, i.e.

$$new\_size = \lceil percentage\_attending \times planned\_size \times (1 + \eta) \rceil \tag{5.1}$$

Here the planned size refers to the size estimated before the term started. The percentage attending is something that can be measured or estimated. Estimations could be made per study, course, course-type or in general and could change each week or day. In this research it is assumed that general attendance changes per week.

5.2 Discrete event simulation

Discrete event simulation concerns the modelling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time (Law [2007]). These time points are the (only) moments in time when an event can occur, here an event is not an educational happening but an instantaneous occurrence that may change the state of the system. Because the notion of event is already used in this research this phenomenon will be called an *occurrence*.

In this research the time points are defined as the start of a time slot and the system state is the room allocation at a particular time. An occurrence is either receiving data from scanners or receiving a reservation request (as given by the Model Architecture, see Figure 2.4, column 3 and 4). The scanner can measure the number of attending people and therefore detect if; the number of people is different than expected, a room is not used while it was expected to be occupied or a room is used while it was expected to be empty. The occurrences in this research are classified as followed;

- **Scanner - new size:** The number of people attending the event differs from the expected number.
- **Scanner - cancel event:** An event was expected in a certain room, but did not occur.
- **Scanner - move event:** An event was expected in a certain room, but did not occur and a request is given to move the event to a later point in time.
- **Scanner - new event:** An unexpected event was detected in a certain room.
- **Reservation bureau - new request:** A reservation request has arrived.

Each time an occurrence occurs the state of the system might change, if and what happens depends on the type of occurrence and is described by the *occurrence routine*. An overview of the occurrence routines is given in Table 5.1.

Table 5.1: Occurrence routines

occurrence	input	immediate action	output
new size	size	update sizes, check utilisation, report bottlenecks	set of future events
cancel	empty room	remove event from room allocation, change room availability	-
move	empty room, new starttime	remove event from room allocation, change room availability , change starttime, report updated event	updated event
new event	new event	change room availability	-
reservation	request	run algorithm, allocate reservation if possible	-

A change in student attendance leads to a *new size* occurrence. Due to these changes it can occur that the currently allocated room is not suitable anymore. As this event has already started it cannot be reallocated, the future

events of the same course can however. It is assumed that student attendance of future events of the same course will remain the same (although no noise is expected). Hence the future events of a course will have a new size of $new\_size = percentage\_currently\_attending \times planned\_size\_before\_term$. If these changes lead to a utilisation larger than 100% or smaller than a predefined *minimum utilisation* bound the future event is marked as a bottleneck event.

When a *cancel occurrence* occurs the event is immediately removed from the room and the room is set back to available again for the duration of the cancelled event. The event is not rescheduled. A *move occurrence* is equivalent to a cancel occurrence, except that the event is rescheduled at a later point in time and marked as a bottleneck (as it has no room yet). A *cancel occurrence* can occur each time slot with probability $P_{cancelation}$ and a canceled event is moved with probability P_{move}.

If an unexpected event is detected (*new event occurrence*) the room is set unavailable for the duration of the event. The event is automatically stopped (read: kicked out of the room) when the next allocated event starts in the room. More information on the creation of new events is given in Section 5.3. A new event can occur each time slot with probability P_{random_event}.

Reservation requests are mostly handled by the reservation bureau (but can be passed to the scheduler team). A reservation is a request for a (specific) room at a predefined time. The event itself occurs during future time slots, anywhere between the next slot and two weeks later. Section 5.4 describes the algorithm used to allocate a reservation.

As can be seen from Table 5.1 the occurrence routines can both perform immediate actions (cancel event, new event, reservation) and collect events which need to be reallocated later (bottleneck events). These collected events are reallocated at a later point in time (see Model Architecture, Figure 2.4 column 3), e.g. at the end of each day or week. One could take into account only the bottleneck events when reallocating, but when taking some of the other future events into account a better solution could be found. Obviously not all future events should and can be reallocated, only a subset, the so called 'flexible' events' are chosen to be reallocated. The flexibility of an event depends among others on its type and size.

Together these immediate actions and periodical reallocations form the Discrete Event Simulation for a Quartile. A flowchart is given in Figure 5.1. Here the time between each reallocation is called a subhorizon, as they divide the time horizon of the Quartile in a number of equally long sub horizons.
Note that information from scanner data and reservation request are created before that simulation starts, this information is not used before the corresponding time slot occurs and therefore not know to the simulation before hand. The arrival time of reservations must therefore also be created before hand. Which is done by picking a random time slot in a predefined time horizon before the reservation is about to occur. The earliest slot that can be picked is of course the start of the Quartile itself.

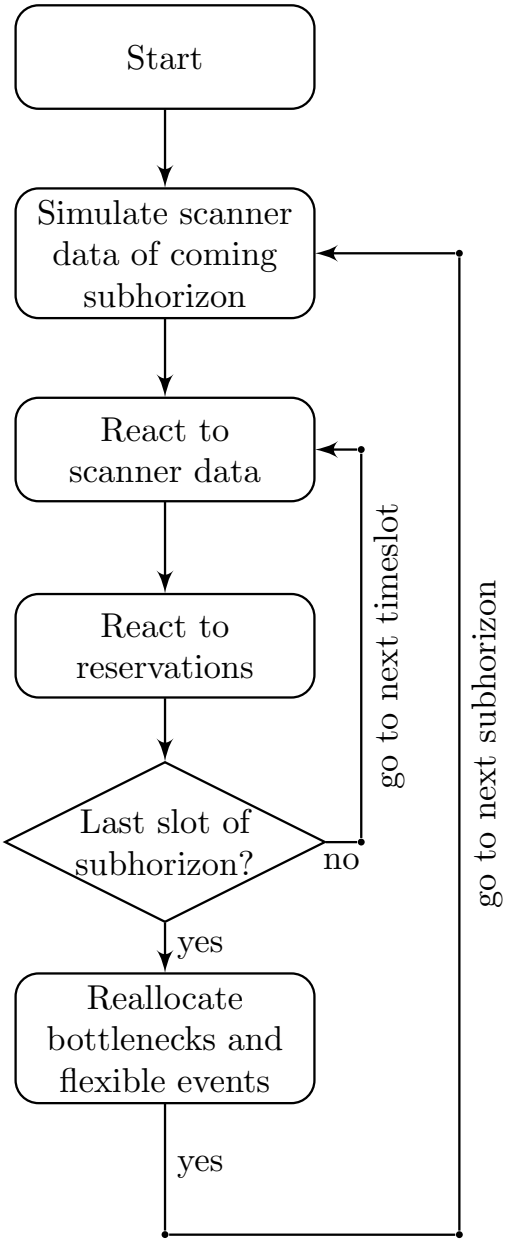


Figure 5.1: Flowchart for the Discrete Event Simulation

5.3 Simulate new events & reservations

To ensure that new events and reservation requests are not completely random historical data is used. The new events are based on the set of events used to create a blueprint, while the reservations are based on data of actual reservations. The new events are often lectures and tutorials, while the reservations are often small project groups. The occurrence routines for both occurrences is also different.

The *new event occurrence* is used to simulate events which, according to the room allocation, should not occur in the given room. This could be events that have moved from some other time or place without notifying the scheduler team, or an event that takes longer than expected and can stay in their allocated room because no other event was planned directly after it, or something completely different. New events appear randomly in available rooms and their duration is adapted such that they do not overlap with currently allocated events to that room.
A reservation request arrives before the actual event occurs (read: wants to occur) and requests a room with certain specifications. If no such room is found the reservation might be moved or canceled.

The creation of a new event or reservation is done as followed;

- Read historical data from file
- Calculate the distribution of certain attributes, e.g. eventtype, date, starttime, duration, size and requested properties.
- Draw attributes from the created distributions and create a new event

The distribution of an attribute is obtained by counting the number of times that the attribute adapts each value in the historical data. I.e. if reservations historically require a room of size six 80% of the time then the probability of creating a reservation of size six is 80%.

5.4 Online algorithm for room allocation

This section gives an overview of the algorithm used to allocate reservations. The algorithm either returns a suitable room or it returns a list of recommended rooms. A suitable room is found by;

1. Gathering all rooms which are available during the time slots the reservation is requesting to occur.
2. Removing all rooms which are too small.
3. Removing all rooms which do not satisfy all the given requirements.
4. Picking the smallest room.

The smallest room is chosen such that bigger rooms remain available for when bigger requests arrive and to minimize the space wastage. If no suitable room is found a list of recommended rooms, in possibly different time slots, is created. Each recommendation is penalised on different aspects;

- Different time slots: current slots, one time step later or one day later, one week later
- Space wastage: room is too big or too small
- One of the requirements is not met.

After all possible room - time combinations are given a score the n best recommendations are returned. The simulation will by default pick the recommendation with the highest score (lowest penalty), but a person might choose a different room.

Chapter 6

Setup and Results: Dynamic allocation

This chapter describes the results of dynamic allocation as described in Chapter 5. Section 6.1 gives an overview of the data used in this research and chosen values for various parameter. Section 6.2 illustrates the effect of changes during term and how dynamic allocation adapts to this.

6.1 Setup

The dynamic room allocations made in this research are based on the actual timetable of the first Quartile of the year 2014-2015 at the UT, just like the static allocations created in Chapter 4. This setup also excludes the evening hours and weekends. As the results in section 4.2.2 concluded that weeks 44 and 45 could not be approximated well these weeks are omitted in this chapter. I.e. the allocation consists of 8 weeks (of 5 days each) and hence spans the time horizon of September 1^{st} up until October 24^{th}.

A dynamic room allocation is always compared to a static room allocation which is made before the term starts. Unless specified otherwise the static room allocation of subsection 4.2.2 (RS) is used, which takes into account room requirements (KPI 3) and space under-/overuse (KPI 6).

During a simulation the static allocation can assign moved events, new events, or reservations (if any) to a room, but cannot reassign existing events (e.g. of which the size has been changed). The dynamic allocation is allowed to reassign existing events, but only at specific periods in time (e.g. at the end of each day or week) and can only reassign flexible events or events which are assigned as a bottleneck. Note that in real life the scheduler team can reallocate bottleneck events during term, but events are (almost) only marked as such when their corresponding room is too small, and not when a room is too big. Hence it mostly corresponds to a static allocation.

In this setup events of type *tutorial*, *colstruction* and *self study* are kept flexible, all other events are fixed (see Table 6.1). Events are reallocated at the end of each week.

Table 6.1: Flexible and fixed event types

flexible	tutorial, colstruction, self study
fixed	lecture, practical, project, exam, presentation, other

As reallocation is done over all future slots (which might consist of multiple weeks) the decomposition technique of section 3.2.1 is used to decrease complexity. The number of sub problems depends on the number of events that can be reallocated/need to be reallocated, and is calculated by:

$$Nsubproblems = \left\lceil \frac{\#\text{Flexible events} + \#\text{Bottleneck events}}{2000} \right\rceil$$

Reallocation will take several objectives into account when searching for an optimal reallocation. To provide good comparison between a static and a dynamic allocation both must use (roughly) the same objectives. The only difference is that, to minimise erratic behaviour (the allocation changing completely every time it is changed), deviation (of the dynamic case) to the static allocation must be taken into account. The deviation (DEV, KPI 4) is given a weight of 5, to show that it has high priority over the other objectives.

To give a clear overview of the influence of Dynamic Room Allocation during the term (and not before) course enrolment is assumed to equal the estimated enrolment. I.e. the probability with which a course changes size ($Pchanged$) is set to zero.

Due to time issues only the occurrences of "new size" could be analysed. The probabilities of cancellation, moving events and random new events are therefore set to zero. Also the noise parameter is set to zero.

The student attendance (see Figure 6.1) is calculated as described in section 5.1 and the percentage with which it changes is based on an educated guess (read: talks with several professors/teachers). Note that this data is neither validated among other teaching personnel nor measured in real life.

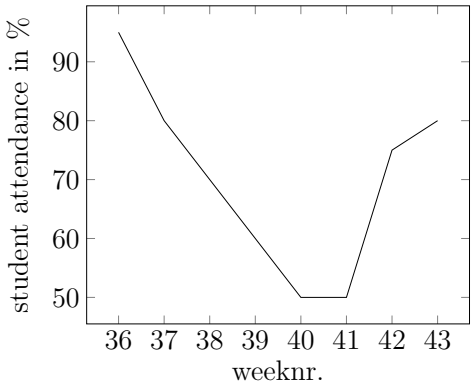


Figure 6.1: Fictional student attendance profile

Future events will be marked as bottlenecks when their estimated utilisation drops below *minimum utilisation* = 70% (unless specified otherwise). When an event is cancelled and moved it will be moved to the same time period in the next week.

6.2 Results

This section illustrates the effect of using Dynamic Room Allocation, compared to using Static Room Allocation. To show the influence of Dynamic Room Allocation we will answer the following questions;

- What is the influence of student attendance on the average utilisation?
- Can Dynamic Room Allocation improve the average utilisation?
- How does the minimum utilisation bound influence the Dynamic Room Allocation?

The first question looks at the influence of student attendance on the average utilisation and shows the disadvantage of Static Room Allocation. The second question shows if Dynamic Room Allocation can improve this average utilisation and the third question shows how the minimum utilisation bound (for marking bottleneck events) has influence on this average utilisation.

6.2.1 Influence of student attendance

This subsection shows the influence of student attendance on the KPI measures space under-/overuse (KPI 6) and utilisation (KPI 5). Apart from a change in student attendance no changes are made to the static allocation, i.e. there is no noise, are no reservations and the probabilities for cancelation, moving and random new events are zero.

Another way to visualise the influence of student attendance is through demand and supply of seat-slots.

Definition 3. (Seat-slot) A seat / time slot pair. A seat-slot can be demanded (for each student during the slots of an event) or supplied (for each seat in an occupied room).

We can therefore define the (estimated) demand per week as the number of seat-slots that are needed in that week, hence the number of (estimated) students for each event times the number of slots the event takes up. And the supply per week as the number of seat-slots that are supplied that week, hence the number of seats in occupied rooms times the slots during which those rooms are occupied.
If average utilisation is high, waste is low and demand and supply will be close to each other. If demand diverges far from supply, waste is high and average utilisation is low.

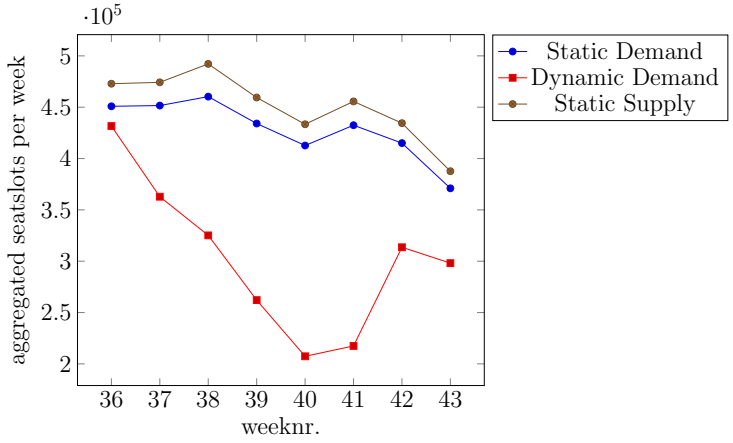


Figure 6.2: Influence of student attendance on demand of seat-slots

Figure 6.2 shows the static demand estimated before term and the dynamic demand which is simulated based on the student attendance profile given in Figure 6.1. The more the student attendance drops, the more estimated and simulated demand differ.

The Figure also shows the seat-slots supplied by the Static Room Allocation, this supply matches the static demand pretty good but when demand becomes dynamic performances decrease quickly. This is also shown in Figure 6.3 which shows the measured utilisation before term (estimated) and during term (simulated). It can be seen that an allocation which performs well based on the estimated data does not necessarily perform well during term (simulated data). This decrease in performance on the KPIs space under-/overuse and utilisation occurs because the simulated data deviates from the estimated data.

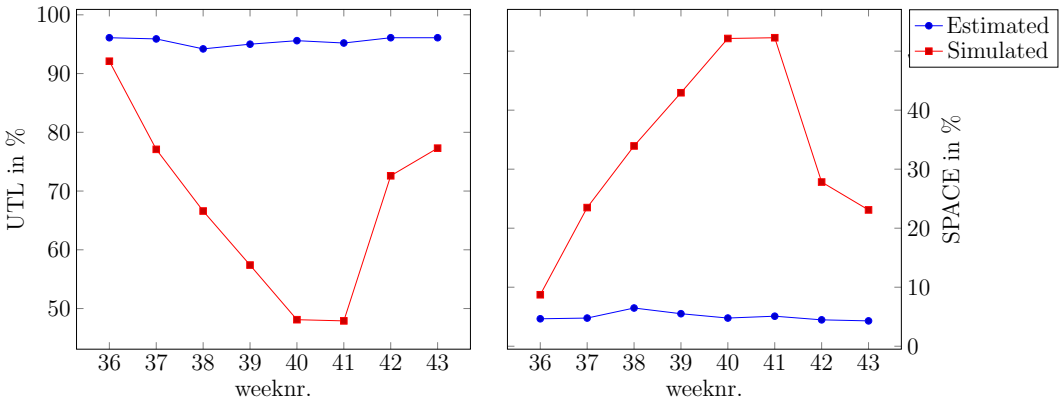


Figure 6.3: Influence of student attendance on utilisation and space under-/over use

6.2.2 Dynamic allocation

This subsection shows the influence of dynamic allocation on the supply of seats. It compares the static case RS (see subsection 4.2.2) and the dynamic case RS.

The dynamic case marks future events as a bottleneck if the estimated utilisation drops below 70%. This estimated utilisation is based on the estimated student attendance, which is assumed to be the same percentage as the attending percentage at the time the event is marked as a bottleneck. I.e. if 50% of the students is attending event i of a certain course-type than it is expected that only 50% will attend event $i+1$ (and other future events) of that course-type. If this leads to an utilisation below 70% event $i+1$ will be marked as a bottleneck.

Hence the dynamic case bases its estimated demand on the actual demand of the current week and the estimated demand for the coming week (and of course not on the future dynamic demand as that is not yet known). Figure 6.4 shows the static and dynamic demand (which were also given in Figure 6.2) and also the demand estimated during term (which is used by the dynamic allocation).

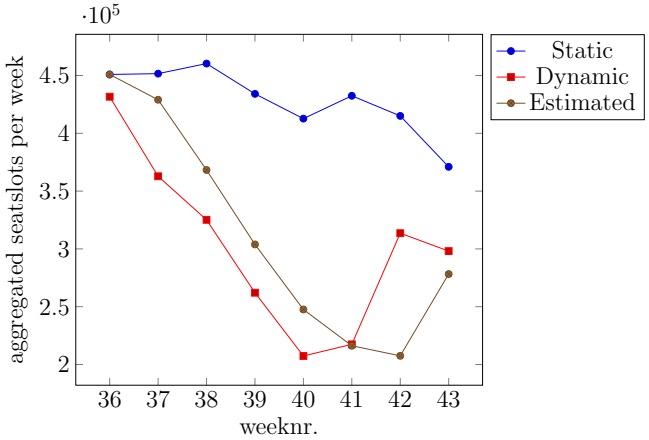


Figure 6.4: static, dynamic and estimated demand for Quartile 1A

The estimated demand looks like a delayed version of the dynamic demand, which makes sense considering that the estimated demand is based on last weeks dynamic demand (except for the first week of course). This works well (or at least much better than the static demand) when student attendance keeps decreasing, but might give some disastrous results when student attendance increases. Then again, teachers will notify when rooms are too small or when they expect them to be too small in the near future. Something they do not do that often for too large rooms.

Anyhow, demand estimation should be improved, but for this real data is needed. For this research the current estimation is sufficient however as it shows if and how dynamic allocation can respond to changes in the demand.

Figure 6.5 shows both static (see also Figure 6.2) and dynamic supply. It can be seen that the dynamic allocation indeed adapts it supply, although it takes a few weeks. Note that this lagging behind (as the estimated demand is based on the previous week) does not pose a problem when student attendance decreases, when it increases however the dynamic allocation could supply less than is really needed. Luckily however staff and students do complain when rooms are estimated to be too small in the future, as opposed to them being too big.

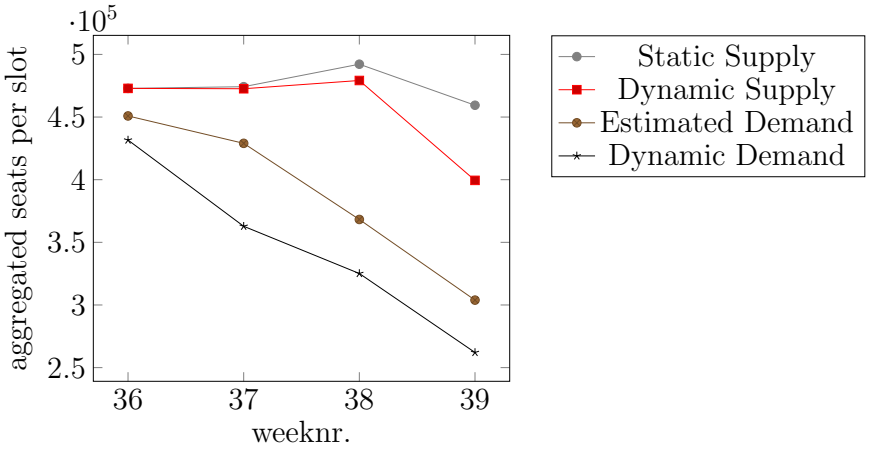


Figure 6.5: Static versus dynamic supply

6.2.3 Influence of minimum utilisation

This subsection shows how a predefined minimum utilisation bound, used to determine events as a bottleneck, influences the average utilisation and other KPIs. This subsection compares the dynamic RS case (as shown in the previous subsection) for various minimum utilisation bounds. The influence on seat-supply for the various cases will be compared. All cases in this subsection use soft constraints S1 and S6 which are both given a weight of one. The dynamic cases also use the deviation soft constraint (constraint S5) with a weight of five.

1. Static allocation (statRS)
2. Dynamic allocation with a minimum utilisation bound of 70% (dynRS70)
3. Dynamic allocation with a minimum utilisation bound of 80% (dynRS80)
4. Dynamic allocation with a minimum utilisation bound of 95% (dynRS95)

Because of time limitations only the first four weeks of Quartile 1A are simulated.

Figure 6.6 shows that the higher the minimum utilisation bound is set the better the dynamic supply matches the estimated demand. Note however that if this estimated demand is off, for example when student attendance suddenly increases, this also means that rooms will be too small quicker. Hence it is important to have a good estimation of the supply.

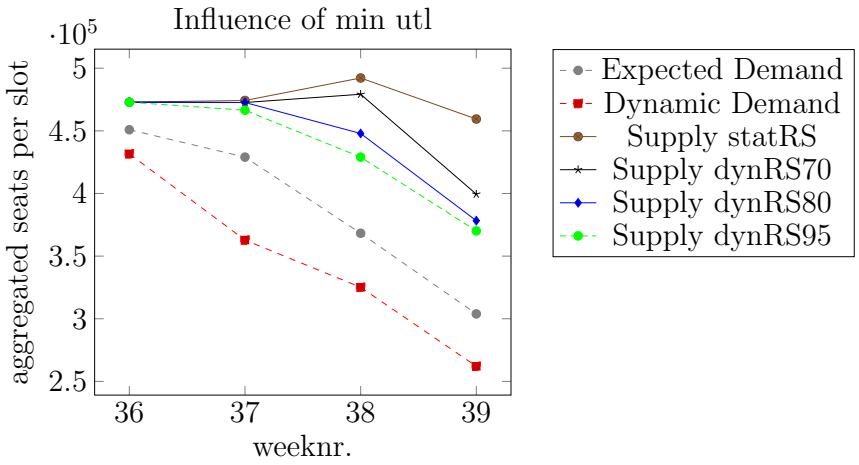


Figure 6.6: Influence of minimum utilisation bound on the supply of seat-slots

Chapter 7

Conclusion

This chapter wraps up the report with a discussion in Section 7.1 and recommendations for future work in Section 7.2. This chapter forms the back matter of the report together with the appendices and bibliography.

7.1 Discussion

The goal of this research was to analyse the effect of usage and utilisation knowledge on the quality of the room allocation. This was done in two step; automatically creating an acceptable room allocation and using this model to dynamically adapt to real-time data.

Chapter 4 showed that the automatically created allocation can approximate the actual room allocation in Quartile 1A, except for the last two weeks, in which exams throw a spanner in the works. While in reality exams can (and often do) share rooms this is not possible in the current model.
Subsection 4.2.1 shows that finding a feasible allocation, given the current assumptions, for Quartile 1A is possible after some minor size adjustments to only three events. This does not necessarily mean that an allocation can be found for the coming Quartile, as it all depends on the input that is given.

Room suitability can be provided in Quartile 1A for at least 89% of the events, and depending on the chosen objectives in some weeks even for 99% of the events (see Figure 4.7). Utilisation and space under-/overuse constraints decrease the percentage of satisfied requirements a bit, but in return improve the utilisation/space under-use a lot (see Figure 4.2). Interestingly the space under-/overuse constraints, which are simpler in design and faster to solve, work better than the utilisation constraints (see Figure 4.7).

Occupation neither seems to work as expected, although that might depend on the input data, which is something for future work.
The course room stability constraint does work as expected and decreases the amount of rooms per course-type and course and meanwhile leads to more satisfied requirements as well.

Hence we can conclude that an acceptable allocation can be created automatically, given the KPIs defined in this report. Although not all KPIs can be controlled easily. More research on the priorities of KPIs and input is needed for this.

Chapter 6 showed that even if a room allocation has good quality before the start of the term, it might give disastrous results during term. To keep up with changes in student enrolment and attendance (demand) dynamic allocation is needed. Even a relatively simple estimation of student attendance and dynamic reallocation shows improvement (see Figure 6.6). To match the demand even better a good estimation on enrolment- and attendance is needed, and hence a lot of (historical) data.

7.2 Future work

To provide the possibility to share rooms and therefore better approximate exam-weeks, research must be done on which event can and which cannot share rooms. The same could be done for events which need multiple rooms at the same time or multiple events which need a certain room after each other. This could be combined with constraints on the distance between these rooms.

Other constraints that could improve the quality of a room allocation, and which would be appreciated a lot by students and teachers alike, are constraints that look at the traveled distance per day. Especially when breaks are short a change between buildings is not desirable.

Improvement of input data, e.g. more detailed room requirements, could also lead to a better room allocation. Requirements on rooms might be different per study or even per teacher and the use of weights could give a more detailed picture on room wishes. The feature to disapprove of certain room (requirements) can also provide new insights.

It is also the question if one should want to satisfy all requirements, and hence have worse utilisation. Should events with a lot of requirements have all their requirements satisfied, while events which are more flexible get worse rooms? What is a fair room assignment? And should it depend on the assignment of last Quartile? All questions that could be researched in future work.

This research provides a way to dynamically reallocate events, and shows that even for poor estimations it can improve the room allocation. More research should be done to improve these estimations on student attendance, and on when events should be reallocated. It is also important to accurately compute the actual student attendance, as measurements might vary through time.
It is also interesting to see how the dynamic allocation responds to real data.

Appendix A

This appendix includes an overview of the Integer Linear Program (ILP) model used in this report and an overview of the Key Performance Indicators (KPIs) denoted in this report.

A.1 ILP model

Sets

This subsection denotes the essential elements needed for the Integer Linear Problem model, these are also the indices of the decision variables.

e	events
$r, r1$	rooms
t	time slots
req	requirements
c	courses
$type$	event types

Parameters

This section denotes the instance-specific data, i.e. input.

E_t	set of events which occur during time slot t
cap_e	size of event e
cap_r	capacity of room r
$conflict_r$	rooms that cannot be used simultaneously with room r
E_{exam}	set of exam events
R_{exam}	set of exam rooms
$require_{e,req}$	1 if event e requires requirement req, 0 otherwise
$sat_{r,req}$	1 if room r satisfies requirement req, 0 otherwise
$utl_{e,r}$	percentages of occupied seats in room r when occupied by event e
$\underline{utl}_r$	target minimum utilisation of room r
T	set of (consecutive) time slots, also denoted as time horizon.
$occ_{r,T}$	average occultation of room r during time horizon T
$\underline{occ}_{r,T}$	target minimal average occupation of room r during time horizon T
dur_e	duration of event e in number of time slots
E_{ctype}	set of events in course c of type $type$
$init_{e,r}$	1 if event e was initially allocated to room r, 0 otherwise
pen_*	penalty for not satisfying soft constraint $*$, indices depend on the constraint

Variables

This subsection denotes the variables used to describe the decisions that must be made by the model.

$x_{e,r}$	1 if event e is allocated to room r, 0 otherwise
y_*	1 if (the set of) constraint(s) $*$ is(/are) unsatisfied, 0 otherwise (indices of y depend on the constraint)
z_*	degree of dissatisfaction, an integer between 0 (satisfied) and 1 (unsatisfied)
$used_{type,r}$	1 if course-type *ctype* uses room r, 0 otherwise
$rooms_{type}$	number of different rooms used by events of course c and event type *type*
$sw_{e,r}$	number of under-/overused seats when allocating event e to room r
$\alpha_{r,T}$	1 if room r is occupied during horizon T, 0 otherwise

Objective function

This subsection denotes the objective function, which measures the quality of solutions.

$$
\begin{aligned}
minimise \sum_{e,req} & y_{e,req} pen_{e,req} && \text{(from constraint S1)} \\
+ \sum_{e,r} & z_{e,r} pen_{e,r} && \text{(from constraint S2)} \\
+ \sum_{r,T} & z_{r,T} pen_{r,T} && \text{(from constraint S3)} \\
+ \sum_{ctype} & (rooms_{type} - 1) pen_{ctype} && \text{(from constraint S4)} \\
+ \sum_{e,r} & y_{e,r} pen_{e,r} && \text{(from constraint S5)} \\
+ \sum_{e,r} & sw_{e,r} pen_{e,r} && \text{(from constraint S6)}
\end{aligned}
$$

Constraints

This subsection denotes the constraints a solution should satisfy, or try to satisfy in case of soft constraints.

Hard constraints

H1. Room per Event	$\sum_r x_{e,r} = 1$	for all e
H2. Event per Room	$\sum_{e \in E_t} x_{e,r} \leq 1$	for all r, t
H3. Room should fit	$x_{e,r} cap_e \leq cap_r$	for all e, r
H4. Room conflicts	$\sum_{e,r1 \in conflict_r} x_{e,r} + x_{e,r_1} \leq 1$	for all e, r
H5. Exam rooms	$x_{e,r} = 0$	for all $e \notin E_{exam}$,$r \in R_{exam}$

Soft constraints		
S1. Room requirements	$\sum_r x_{e,r} require_{e,req} \leq y_{e,req} + \sum_r sat_{r,req}$	for all e, req
S2. Minimum utilisation	$utl_{e,r} + z_{e,r} \geq x_{e,r} \underline{utl}_r$	for all e, r
S3. Min. average occupation		
a)	$occ_{r,T} + z_{r,T} + \alpha_{r,T} \geq \underline{occ}_{r,T}$	for all r, T
b)	$occ_{r,T} \leq 1 - \alpha_{r,T}$	for all r, T
S4. Rooms per course type	$\sum_{e \in E_{ctype}} x_{e,r} \leq \lvert E_{ctype} \rvert used_{ctype,r}$	for all $ctype, r$
S5. Deviation	$init_{e,r} - x_{e,r} \leq y_{e,r}$	for all e, r, s.t. $init_{e,r} = 1$
S6. Space over-/underuse	$sw_{e,r} = \lvert cap_r - cap_e \rvert dur_e x_{e,r}$	for all e, r

A.2 List of KPIs

This section gives an overview of the Key Performance Indicators (KPIs) used in this report. For each KPI a definition is given an formulas to calculate the KPI.

KPI 1 (Allocations) The percentage of events that is allocated.

$$Allocations = \frac{\sum_{e,r} x_{e,r}}{|E|} \tag{KPI 1}$$

Where $|E|$ denotes the number of events in the problem.

KPI 2 (Misfits) The number of events that are allocated to a room that is too small.

$$Misfits = \#\text{unsatisfied H3 constraints} \tag{KPI 2}$$

KPI 3 (REQ) The percentage of room requirements that are met.

$$REQ = 1 - \frac{\sum_{e,req} y_{e,req}}{|E||req|} \tag{KPI 3}$$

Where $|E|$ denotes the number of events in the problem and $|req|$ the number of possible room requirements.

KPI 4 (DEV) The percentage of room-event pairs that differ (i.e. are modified) compared to a given room allocation.

$$DEV = \sum_e deviated_e \tag{KPI 4}$$

Where $deviated_e$ equals one if $\sum_r y_{e,r} \geq 1$ and zero otherwise (see equation 3.5).

KPI 5 (UTLn) The average utilisation of all used rooms (with size of at least n) during a given time horizon.

$$UTLn = \frac{\sum_{r \in R_n} utl_{r,T} cap_r}{\sum_{r \in R_n} cap_r} \tag{KPI 5}$$

Where R_n denotes the set of rooms with size of at least n. For the calculation of $utl_{r,T}$ see equation 3.2.

KPI 6 (SPACE) The percentage of unused/overused seats in occupied rooms during a given time horizon.

$$SPACE = \sum_{e,r} sw_{e,r} \tag{KPI 6}$$

Where $sw_{e,r} = |cap_r - cap_e| dur_e x_{e,r}$.

KPI 7 (OCCn) The proportion of rooms (with size of at least n) that have been occupied during a given time horizon.

$$OCCn = \frac{\sum_{r \in R_n} occ_{r,T}}{|R_n|} \tag{KPI 7}$$

Where R_n denotes the set of rooms with size of at least n and $|R_n|$ the number of rooms in that set. For the calculation of $occ_{r,T}$ see equation 3.3.

KPI 8 (ROOM) The number of rooms that have been occupied at least once during a given time horizon.

$$ROOM = \sum_{r} occ_{r,T} \tag{KPI 8}$$

Where T contains all the time slots of the given day.

KPI 9 (CROOM) The number of different rooms that are used for a given course-type.

$$CROOM = \frac{\sum_{ctype} rooms_{ctype}}{|ctype|} \tag{KPI 9}$$

Where $|ctype|$ denotes the number of course-types. For the calculation of $rooms_{ctype}$ see equation 3.4.

Bibliography

S. Abdennadher, M. Saft, and S. Will. Classroom assignment using constraint logic programming, 2000.

D. Alan, A. Bakker, J. Muis, E. van de Pieterman, and G. Teerenstra. Realtime application for schedulers of time-constrained assets - iito group 7. Manuscript in preparation, 2015. URL `iito.nl`.

M. Ayob and A. Malik. A new model for an examination-room assignment problem. *International Journal of Computer Science and Network Security*, 11(10):187–190, 2011.

D. Baars, A. van den Berg, and P. Höner. Monitoring ut room utilisation - business case development for it-projects, June 2014.

C. Beyrouthy, E.K. Burke, D. Landa-Silva, B. McCollum, P. McMullan, and A.J. Parkes. The teaching space allocation problem with splitting. In *Practice and Theory of Automated Timetabling VI*, pages 228–247. Springer Berlin Heidelberg, 2007a.

C. Beyrouthy, E.K. Burke, D. Landa-Silva, B. McCollum, P. McMullan, and A.J. Parkes. Improving the room-size profiles of university teaching space. Technical report, Technical Report). School of Computer Science, University of Nottingham, 2007b.

C. Beyrouthy, E.K. Burke, B. McCollum, P. McMullan, and A.J. Parkes. University space planning and space-type profiles. *Journal of Scheduling*, 13(4):363–374, 2010.

E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280, 2002.

E.K. Burke and D.B. Varley. Space allocation: An analysis of higher education requirements. In *Practice and Theory of Automated Timetabling II*, pages 20–33. Springer, 1998.

E.K. Burke and D.B. Varley. Automating space allocation in higher education. In *Simulated Evolution and Learning*, pages 66–73. Springer, 1999.

E.K. Burke, P. Cowling, D. Landa-Silva, and B. McCollum. Three methods to automate the space allocation process in uk universities. In *Practice and Theory of Automated Timetabling III*, pages 254–273. Springer, 2001.

E.K. Burke, J. Mareček, A.J. Parkes, and H.Rudová. Decomposition, reformulation, and diving in university course timetabling. *Computers & Operations Research*, 37(3):582 – 597, 2010.

M.W. Carter and G. Laporte. Recent developments in practical course timetabling. In Edmund Burke and Michael Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin Heidelberg, 1998.

M.W. Carter and C.A. Tovey. When is the classroom assignment problem hard? *Operations Research*, 40(1-supplement-1):28–39, 1992.

A.A. Constantino, W.M. Filho, and D. Landa-Silva. Iterated heuristic algorithms for the classroom assignment problem. In *Practice and Theory of Automated Timetabling VIII*, pages 152–166. Queen?s University Belfast, 2010.

A.W. Dijksterhuis. Kwaliteit roosters universiteit twente - onderzoek naar het meetbaar maken van de prestaties van het rooster op universiteit twente. http://essay.utwente.nl/65103/, May 2014. Bachelor thesis.

A. Elloumi, H. Kamoun, B. Jarboui, and A. Dammak. The classroom assignment problem: Complexity, size reduction and heuristics. *Applied Soft Computing*, 14: 677–686, 2014.

E.W. Hans, M. Van Houdenhoven, and P.J.H. Hulshof. A framework for health care planning and control. Memorandum 1938, Department of Applied Mathematics, University of Twente, Enschede, February 2011. http://eprints.eemcs.utwente.nl/19571/.

E.C. Kerrigan and J.M. Maciejowski. Soft constraints and exact penalty functions in model predictive control. In *Control 2000 Conference, Cambridge*, 2000.

D. Landa-Silva. *Metaheuristic and multiobjective approaches for space allocation*. PhD thesis, University of Nottingham, 2003.

A.M. Law. *Simulation Modeling & Analysis*. McGraw Hill, 4th edition, 2007.

H. Martinez-Alfaro and G. Flores-Teran. Solving the classroom assignment problem with simulated annealing. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 4, pages 3703–3708. IEEE, 1998.

H. Martinez-Alfaro, J. Minero, G.E. Alanis, N.A. Leal, and I.G. Avila. Using simulated annealing to solve the classroom assignment problem. In *ISAI/IFIS 1996. Mexico-USA Collaboration in Intelligent Systems Technologies. Proceedings*, pages 370–377. IEEE, 1996.

S.A. MirHassani and F. Habibi. Solution approaches to the course timetabling problem. *Artificial Intelligence Review*, 39(2):133–149, 2013.

A.E. Phillips, H. Waterer, M. Ehrgott, and D.M. Ryan. Integer programming methods for large-scale practical classroom assignment problems. *Computers & Operations Research*, 53:42–53, 2015.

H. Punt, L. Woud, J. Sevens, J. Pasman, and J. Nijhof. Onderwijsprotocol 1314 versie 1.0, 2012.

A. Schaerf. A survey of automated timetabling. *Artificial intelligence review*, 13(2): 87–127, 1999.

H. Westerik. Key performance indicators voor roosters v1.0, 2013.