



Aperiodic and Group-Sensitive Constraints on Curriculum-Based Timetabling

Bachelor Thesis

by **Philipp Günther**

To obtain the rank „Bachelor of Science“ (B. Sc.)
in Computer Science (Informatik)

First Referee: Prof. Dr. Rolf Niedermeier
Second Referee: Prof. Dr. Thorsten Koch
Advisers: Dr. André Nichterlein, Leon Kellerhals

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbstständige und eigenständige Anfertigung versichert an Eides statt:

Berlin, den _____

Abstract

We look at newly found demands and requirements of universities employing reform study programs regarding timetabling. Whereas classic study programs usually repeat the same schedule of courses every week, reform programs utilize many smaller, aperiodic courses and have a student-group-centric approach to planning. Tackling the problem of computing such a non-periodic schedule is the main focus of this thesis.

We present a new model called `APERIODIC CURRICULUM-BASED TIMETABLING` which supports aperiodic courses and student groups as an extension of the known `CURRICULUM-BASED TIMETABLING`. A hypergraph is used to encode scheduling conflicts within the search space. Because of the aperiodicity of most courses, the search space becomes much larger. Therefore, some ideas for splitting the problem into smaller pieces are explored and a connection between the employed hypergraph and `HITTING SET` is made.

Finally, an integer programming approach is presented that solves the problem heuristically in two steps. We show that the decision problem version of the first step is NP-complete. Further, we analyze the performance of the proposed heuristic on real-world data of a medical university employing a reform program. Using the two-step heuristic schedules containing aperiodic alongside periodic courses can be computed within a relatively short time.

Zusammenfassung

Während klassische Studiengänge vorwiegend aus Kursen mit wöchentlich wieder auftretenden Terminen bestehen, setzen sich Modellstudiengänge mehrheitlich aus Einzelterminen zusammen. Weiterhin gibt es deutlich mehr Kurse, die in Kleingruppen gelehrt werden. Diese Arbeit setzt sich mit der Errechnung eines Stundenplans mit derartigen Anforderungen auseinander.

Wir stellen ein neues Modell namens `APERIODIC CURRICULUM-BASED TIMETABLING` vor, welches Studentengruppen und sowohl periodische als auch aperiodische Kurse unterstützt. Dabei wird das bekannte Problem `CURRICULUM-BASED TIMETABLING` unter anderem um Semesterwochen erweitert. Wegen der aperiodischen Kurse wächst jedoch der Suchraum enorm an und es werden verschiedene Ansätze behandelt um das Problem in kleinere Stücke zu unterteilen. Konflikte innerhalb des Suchraumes sind mit Hilfe eines Hypergraphen modelliert, von dem eine Verbindung zu `HITTING SET` hergestellt wird.

Letztlich wird ein ganzzahliges lineares Programm vorgestellt, das das Problem heuristisch in zwei Schritten löst. Der erste Schritt der Heuristik wird in seiner Version als Entscheidungsproblem als NP-vollständig ermittelt. Anschließend wird die Heuristik mit realen Daten einer medizinischen Universität mit Modellstudiengang getestet und ihre Laufzeit untersucht. Die vorgestellte Heuristik errechnet innerhalb relativ geringer Laufzeit für die Anforderungen passende Studienpläne.

Contents

1	Introduction	7
1.1	Terminology	8
1.2	User Interfaces	9
1.3	Outline	11
2	Model & Key Observations	15
2.1	Notation	18
2.2	Assigning Organizations to Classes	20
2.3	Definition of the Problem	22
2.3.1	Conflicts	22
2.4	Key Observations	24
2.4.1	Size of the Conflict Graph	24
2.4.2	Connection to Hitting Set	25
2.4.3	A Simple Reduction Rule	26
3	Proposed Heuristic	29
3.1	Step 1	29
3.1.1	NP-completeness	31
3.1.2	Integer Linear Program	36
3.2	Step 2	37
3.2.1	Integer Linear Program	38
3.2.2	Room Set Contraction	40
3.3	Performance & Benchmarks	41
3.3.1	Real World Implementation Differences	41
3.3.2	Benchmarks	43
4	Conclusion & Outlook	47
	Literature	49

1 Introduction

Automatically generating timetables for schools and universities is an active field of research [KS13; Sch99; Wer85]. Under the category of *Educational Timetabling* a number of problems exist that address different aspects of timetabling. For universities in general, a distinction is made between *Examination Timetabling* and *(University) Course Timetabling*. For the latter, one distinguishes between *Curriculum-Based Timetabling (CBT)* and *Enrollment-Based Timetabling (EBT)*.

Both *Course Timetabling* problems and even *Highschool Timetabling* have in common that they produce a weekly schedule that repeats throughout the whole teaching period. Examination Timetabling, however, produces a schedule for specific dates and has different constraints. One major constraint is the desired amount of free time between examinations. Per student and exam one day is in general to be kept clear. In Course Timetabling, however, it is desirable to schedule few breaks for the students.

When talking about CBT courses have to be scheduled conflict-free for every curriculum. The curricula are defined by the educational institution in order to enable students to study programs within reasonable time periods. In EBT, however, it is known which student is enrolled in which courses beforehand. Usually the enrollment information is collected from the students themselves. Therefore, a timetable is to be constructed so that every student can attend all courses in which she or he is enrolled.

Universities constructing all their schedules using automated methods generally use a combination of CBT, EBT, and Examination Timetabling as depicted by Kristiansen and Stidsen [KS13, p. 8]: First, a rough schedule is constructed using CBT based on the known curricula. Contained in this step are mostly lectures that are attended by larger numbers of students. For planning smaller courses, such as tutorials, some form of EBT is used as a second step, where students enroll at the beginning of the semester. Usually these courses are held in small groups requiring a different set of rooms than big lectures. Therefore, smaller rooms are often blocked in the first step using CBT to reserve time for the smaller courses later on. After that, the examinations are planned using Examination Timetabling.

New Requirements The approach described above starts falling apart when the number of small courses starts to prevail. Sometimes the enrollment data cannot be obtained or it becomes unclear how many resources have to be blocked during CBT so that it will be possible to find a feasible timetable during EBT.

What makes both Course Timetabling versions unfit as well is the limitation to only be able to construct weekly schedules. There are two possibilities to deal with events that do not run for the whole semester, such as block seminars, congresses and so on. The

1 Introduction

first is to naïvely schedule these events as normal courses. This results in the assigned rooms being otherwise empty for the rest of the semester and is thus inadequate. The second approach is to, again, block some rooms while constructing the weekly schedule and later placing these events by hand. Being manual labor this is only acceptable if the number of such events is low. So for universities with many non-weekly events, both approaches are undesirable.

New Model Many small and non-periodic courses motivate a new problem definition. First and foremost, non-periodic courses need to be handled well, meaning they cannot block a combination of room and time for the whole teaching period while only using a small portion thereof. Second, a different approach for small courses is needed which still enables to maintain conflict-free curricula per student. Hence, we introduce *Aperiodic Curriculum-Based Timetabling (Aperiodic CBT)* in this thesis.

Both issues are stressed by recent advancements made in curricula development. Some study programs have been reformed to deviate from traditional programs [SLH08; Cha]. These new reform curricula almost exclusively consist of aperiodic courses and have many courses that students have to attend in small groups. This is because mostly medical institutions employ these new curricula, where for instance bedside teaching is a common occurrence. There the students are taught next to a patient’s hospital bed, thus requiring small sets of students.

Also a problem for medical institutions is the assignment of teachers or lecturers to courses. Most organizations that are involved in teaching students also have to handle patients. Therefore, who exactly is going to teach a class is often decided on the spot by who is available. Hence, in contrast to CBT and EBT, teachers are not assigned to courses beforehand. Instead each course is assigned a set of organizations that are suitable for teaching the subject. Therefore, Aperiodic CBT does not guarantee conflict-free schedules for teachers. Still, in order not to overload the organizations at any given moment, the work load of organizations can be limited by specifying capacities. These capacities state how many minutes of teaching is acceptable per week, per day and in parallel for every organization. The old model of teachers or lecturers with conflict-free schedules can still be modeled by setting the parallel capacity to one.

1.1 Terminology

In the following, we describe the mentioned reform programs further. We will refer to a curriculum as being a reform program when it contains a mix of periodic and aperiodic courses. Periodic courses are usually referred to as lectures in CBT, meaning that they generally run over the course of the whole lecture period in a weekly manner. Aperiodic courses, however, have a single appointment somewhere during the lecture period.

The goal of this thesis is to schedule some smaller traditional programs alongside a specific reform curriculum that almost exclusively consists of aperiodic courses. The reform program includes seminars, single lectures and talks, practical training courses, bedside teaching and more teaching formats with diverse requirements.

In contrast to traditional programs, most of the courses are held in small groups. Only some lectures are held together for all students of a semester. When a course is not held together for all students, several instances of a course, so-called classes, have to be held and planned.

To specify which students attend a class together, all students of a semester are first partitioned into base groups. A base group is an inseparable set of students. These base groups are then combined to larger groups depending on the teaching format. For a lecture that is held together for all students of a semester, all base groups of the semester form one large group. For very small courses, however, each base group could form its own group. Depending on the number of students per semester, the base groups are partitioned into several cohorts. A cohort is a set of base groups in the same semester.

All courses are organized in modules. For aperiodic courses, the modules can be divided into submodules, each having a certain length given in days. Then, an ordering of these submodules is stated per cohort, which results in a certain time frame for every course per combination of submodule and cohort.

By dividing the students of a semester into cohorts and specifying different submodule orderings per cohort, peak loads for organizations can be smoothed. By only sending one cohort at a time instead of all the students of a semester to a specific organization, its load can shrink from impossible to manageable.

Every base group has a fixed semester and cohort and therefore a well defined submodule ordering. Normally, groups would not be allowed to span base groups of multiple cohorts because their submodule orderings could be different. It is still desirable, however, to be able to form groups spanning multiple cohorts when the corresponding submodule orderings allow for it. Therefore, the orderings are analyzed and groups spanning cohorts are allowed where the time frames of several cohorts align.

We will present some illustrations for the introduced concepts shortly.

1.2 User Interfaces

In the following we will provide some insights into the user interfaces used to enter the data for the reform program within *Moses* [Mata]. **Figure 1.1** depicts submodule orderings of a single semester. These can be entered after all modules and their submodules have been specified.

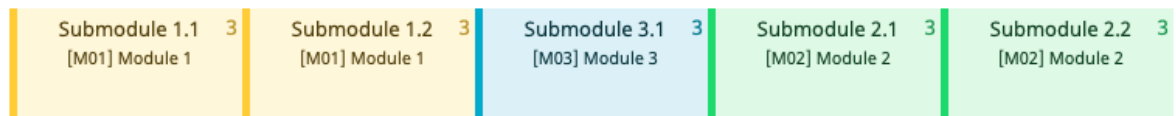
After all courses have been entered and the curricula are defined, the submodule orderings are analyzed for overlaps. For courses, where submodule orderings overlap, groups can be formed spanning multiple cohorts. **Figure 1.2** shows how the base groups are compound to different groups, sometimes spanning cohorts, sometimes not.

Capacities for organizations can be entered as seen in **Figure 1.3**.

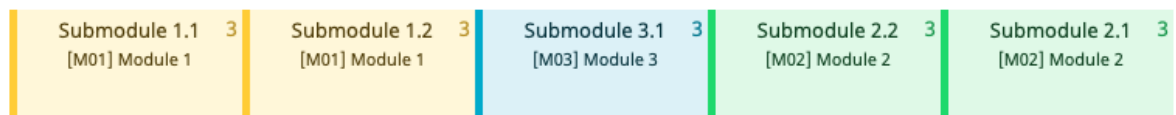
Starting out as a research project at *Technische Universität Berlin* in 2003 and transitioning into a private business at *MathPlan GmbH* [Matb] in 2013 *Moses* has since expanded to three of the biggest technical universities in Germany. It has been rolled out at *Technische Universität Berlin (TUB)* in 2003 [TUB], then at *RWTH Aachen (RWTH)* in 2013 [RWT] under the name *Carpe Diem!*, and at *Technische Universität*

Rotationen

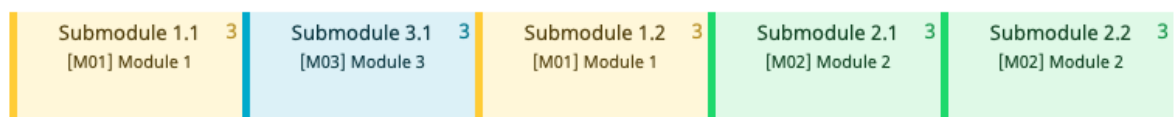
Rotation 1



Rotation 2



Rotation 3



Rotation 4

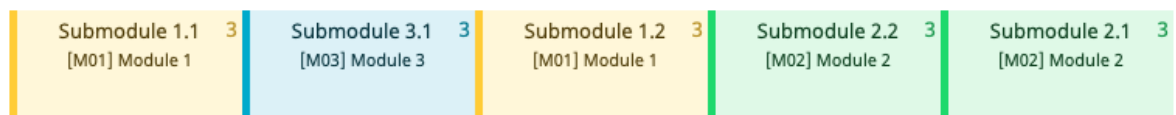


Figure 1.1: A screenshot of four submodule orderings in Moses. Recall that a submodule is a collection of aperiodic courses used to arrange a time-wise ordering. Each box represents a submodule and the different colors mark the modules. All students of a semester are partitioned into the same number of cohorts as submodule orderings exist. For all students participating in rotation one for example all courses that are part of submodule 1.1 have to be completed before any course of submodule 1.2 is able to start. The same goes for students within rotation two. For students of rotations three and four, however, submodule 3.1 follows after submodule 1.1.

München (TUM) in 2014 [TUM]. *TUB* uses the full suite of CBT, EBT and Examination Timetabling, whereas *RWTH* uses CBT and Examination Timetabling [RWTH]. *TUM* uses the Examination Timetabling exclusively [TUM]. The goal of this thesis is to implement the newly described algorithm within Moses and reach universities employing reform study programs, especially medical universities.

In addition to an implementation of the newly formulated problem described in this thesis, there exist real-world implementations of other already mentioned timetabling solutions within Moses. CBT has been implemented and shown by Lach, Lach, and Zorn [LLZ16b], alongside EBT by Höner and Lach [HL16] and Examination Timetabling by

Lach, Lach, and Zorn [LLZ16a].

1.3 Outline

This thesis includes an abstract description and problem definition for APERIODIC CURRICULUM-BASED TIMETABLING in [Chapter 2](#). First, required notation is listed before giving the exact problem definition. Afterwards, some key observations about the conflict graph that is part of the problem are given. A connection of the conflict graph to HITTING SET is investigated. [Chapter 3](#) proposes a 2-step ILP heuristic solving APERIODIC CBT. A polynomial-time reduction from 3-SAT to the first step of the heuristic is given and therefore proven that Step 1 is NP-hard. Afterwards, the performance of the heuristic is analyzed. Despite the NP-hardness of step 1, the runtime is found to be relatively small for the tested real-world instance. Some approaches for further research are given in [Chapter 4](#).

1 Introduction

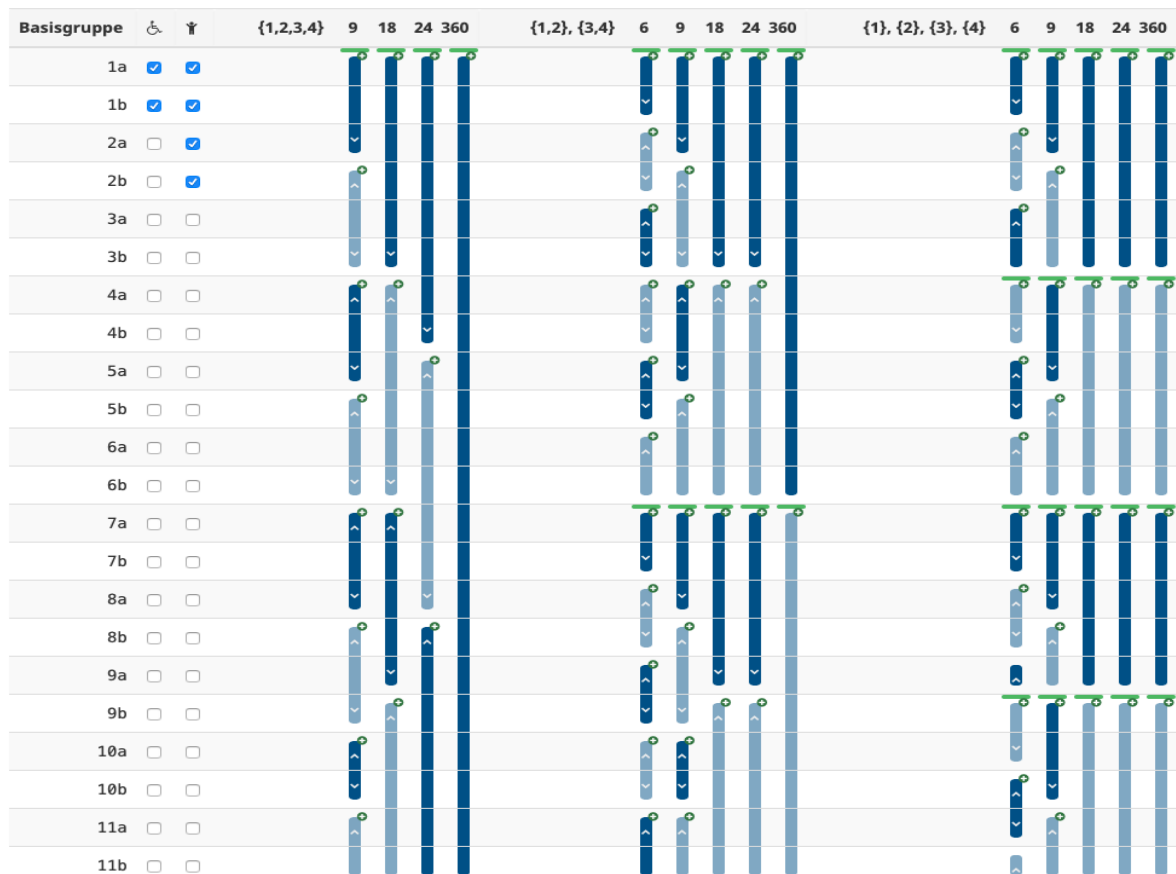



Figure 1.2: A screenshot of groups in Moses. The table shows base groups on the y-axis along with a checkbox for determining whether only accessible rooms or family-friendly times can be scheduled. Each base group is representative of three students. Depending on the teaching format, different numbers of base groups form a group together. Optimal group sizes are displayed in the header along the x-axis. Groups are displayed using vertical bars that span the base groups. Since four submodule orderings exist (see Figure 1.1) the base groups are partitioned into four cohorts, whose borders are marked by green horizontal bars. If the placing of submodules overlap the borders are neglected. All orderings overlap on submodule 1.1 for example. Therefore, lectures within submodule 1.1 can be held for all cohorts of the semester together. This can be seen in the leftmost column titled 360. There all 22 pictured base groups form a single group together. For submodules 2.1 and 2.2, however, no adjacent orderings overlap. Hence all borders between orderings have to be respected. If we now take bedside teaching for example, where optimally six students form a group, we can see in the rightmost column labeled 6 that for the first two cohorts two base groups form a group. For the last two cohorts, however, the partitioning does not work out. Thus, respectively at the end of each cohort a group of a single base group exists, covering only three students each.

Beginn Kernzeit	Ende Kernzeit	Maximale Anzahl paralleler Termine
<input type="text" value="09:00"/>	<input type="text" value="16:00"/>	<input type="text" value="4"/>
Maximale Stunden Lehre pro Tag	Maximale Stunden Lehre pro Woche	
<input type="text" value="6"/>	<input type="text" value="25"/>	

Sperrintervalle

Als Administrator der Stunden- und Raumplanung können Sie Sperrintervalle setzen, in die für diese Einrichtung keine Lehrveranstaltungen eingeplant werden.

Zeitraum	Semesterwochen	
Mo 08:00 - Fr 08:30	1, 3, 5, 7, 9, 11, 13, 15	 

[+ Neues Sperrintervall anlegen](#)

Semesterwochenpräferenz

Bitte wählen Sie Ihre Präferenzen bezüglich Lehre in den folgenden Semesterwochen aus. Durch einen Klick auf eine der Wochen markieren Sie diese als **bevorzugt**, ein weiterer Klick darauf erklärt sie zur für Lehre **unerwünscht** Woche. Weiß hinterlegte Wochennummern repräsentieren eine neutrale Haltung gegenüber Lehre in der betreffenden Woche.

Bitte beachten Sie: **Wochen außerhalb der Vorlesungszeit** sind ausgegraut. Hier findet (für die meisten Studiengänge) standardmäßig zwar keine Lehre statt, Sie können sich aber trotzdem entscheiden Präferenzen für diesen Zeitraum angeben.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
21	22	23	24	25	26																

[✖ zurücksetzen](#)

Figure 1.3: A screenshot of organizational capacities in Moses. Each organization can state their available times. Also, a maximal number of parallel classes, a maximum number of hours per week and a maximal number of hours per day can be added. Finally, entire time intervals can be blocked altogether and preferences for weeks can be stated. In this example the organization states that its teaching personnel is only available between 9am and 4pm. It can teach a maximum of 4 classes in parallel and 6 hours per day. In a week the organization can teach 25 hours in sum. On every week with an odd number Monday 8am to 8:30am is blocked. The organization prefers teaching in weeks two to four and wants to avoid teaching in weeks 11 to 13 of the semester.

2 Model & Key Observations

In this section, we will give a more formal description of the decision problem and portrait some observations. [Section 2.1](#) will give an overview about which symbols are used to describe the used sets of resources and functions. Then, [Section 2.2](#) will cover a preprocessing step, before [Section 2.3](#) will formally describe the decision problem. Finally, in [Section 2.4](#) we will discuss the observations made.

Input

Recall that there are several courses per semester, which consist of classes. Every class of the same course covers the same topics, only for another set of students. Further, each class can be held in several sessions for traditional periodic courses or one session for a fully aperiodic course. The planner specifies the number of sessions and their lengths in minutes. If all courses were to be held once every week of the semester, then the problem would be suitable to be solved using classic *Curriculum-Based Timetabling (CBT)*.

For example, consider a single semester with two courses. One course is a lecture with enough capacity for all students. The other course is a smaller seminar which requires two classes. Then we would have three classes in total: one for the lecture and two for the seminar. Both classes of the seminar would cover exactly the same topics but for a different set of students.

Groups How many classes are needed for every course is dictated by its specified groups. Students are organized in disjoint base groups. All students of the same base group attend the same classes. Depending on the format of the course, several of these base groups form a larger group, so a group consists of a set of base groups. For every course, it is then specified which set of groups has to attend it. There are always exactly as many classes per course as groups.

So for our example, we would need at least two base groups. For the smaller seminar, each base group would also form a group. However, for the lecture, a separate group needs to exist, consisting of both base groups. This makes a total of three groups for our example.

Time The time frame that a course can be scheduled in is given per combination of course and group as a set of eligible periods. These sets can be different within the same course. This makes it possible to offer courses in different orderings for different groups. Thus the assignment of a group to a class yields a certain time window, providing a set of eligible periods. This will become important later on in Step 1 of our proposed heuristic ([section 3.1](#)).

2 Model & Key Observations

Again, for our example, the lecture is supposed to run over the whole semester and only has one group. Therefore, the set of eligible periods is only specified once, whereas for the seminar, a differing set of eligible periods is given for every group. For one group, the first half of the semester is considered eligible and for the other group only the second half. This is done to spread the load of teaching among the organizations.

In the end, each class needs to be assigned one of the eligible periods to define the time it takes place. A period can contain several sessions. A session is a set of

consecutive time slots. In addition to time slots in CBT, where time slots are a tuple of a day and a time interval, we need to keep track of the week within the semester. All time slots need to be disjoint. For fully aperiodic courses, all eligible periods have exactly one session whereas for periodic courses, the periods contain several.

So each eligible period of our periodic example lecture has as many sessions as the number of weeks the semester runs. See [Figure 2.1](#) for an example of an eligible period for the lecture. The lecture needs to take place in every week. However, the seminar is an aperiodic course, so its eligible periods only contain a single session.



Figure 2.1: Visualization of a single periodic period. Every colored box represents a consecutive session of a single period. Every session can consist of several time slots. We call such a period periodic because it has more than one session. This period takes place on Monday starting at 10:00 and lasting two hours. It occurs in all 14 weeks.

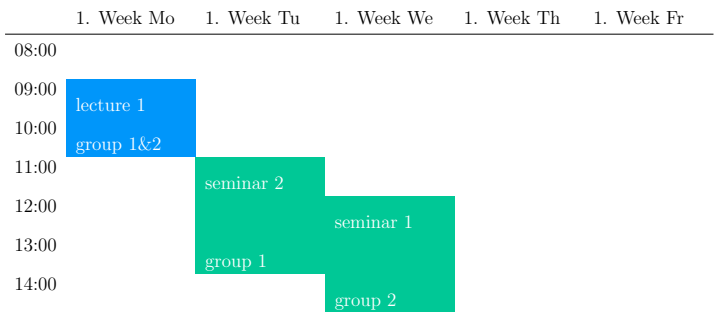


Figure 2.2: Possible timetable for the example

Organizations For assigning organizations to classes, a set of combinations of organizations is given per course, together with the number of classes they are obliged to teach. For each of these combinations of organizations, a set of eligible rooms along with preferences can be stated.

[Figure 2.3](#) gives an overview of how assignments depend on each other and what is assigned to each class. The lecture can only be taught by one specific organization because it covers a special topic. Then the set of eligible combinations of organizations consists of a single element, which is a set itself consisting of only this specific organization. Also

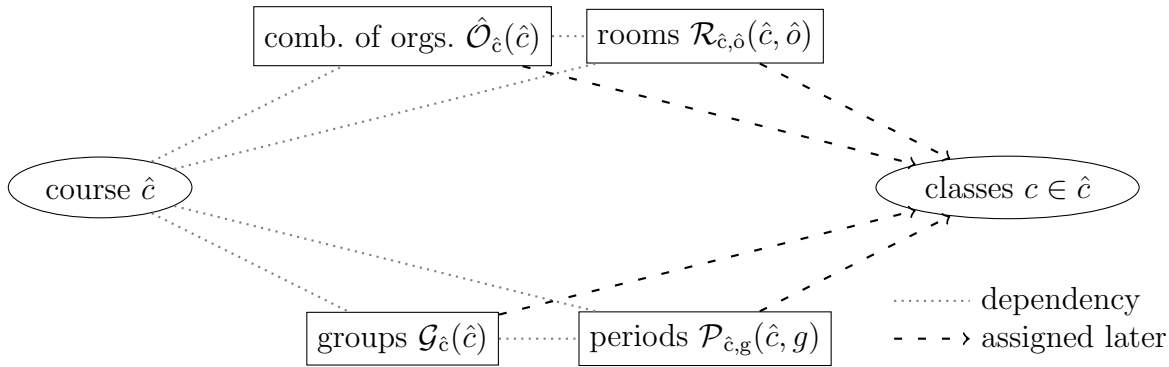


Figure 2.3: Interdependencies of resources and assignments for a single course. Dotted lines show dependencies, and dashed arrows visualize assignments. Every class belongs to exactly one course and needs to be assigned a combination of organizations, room, group and period. Eligible combinations of organizations depend on the course only, whereas the set of eligible rooms depends on both the course and the combinations of organizations. On the other side, eligible groups only depend on the course, and the set of eligible periods depends on both the course and the assigned group. Omitted are possible restrictions on the periods that are given by limited availabilities of the assigned room and organizations.

only one special room is eligible. Therefore, the set of eligible rooms is a set of the specific required room. The seminar, however, covers an interdisciplinary subject and therefore needs two organizations for every class. Also all rooms of a certain minimum size are eligible.

Capacities To keep the amount of teaching in check, each organization can indicate how many classes they can teach in parallel and how many minutes of teaching they can deliver per day and week independently of the course. Also, time slots can be prioritized or outright blocked if internal restrictions demand it.

For example if the only organization required for the lecture absolutely cannot teach on Tuesday afternoons, the lecture cannot be scheduled then.

A resulting timetable for the example lecture and seminar could look like Figure 2.2 in the first week of the semester.

Constraints

Obviously, every class needs to take place at some eligible period, in an eligible room, with an assigned group and a combination of organizations. Also, every group obligated to attend a course needs to be assigned to exactly one class belonging to the course.

All other constraints are conflicts among a set of assignments. As with classic CBT, no room double bookings are allowed. So no two classes with overlapping periods must take place in the same room. Further, we need to guarantee that no student has to attend two classes at the same time. Hence, no two classes with overlapping groups must take place in overlapping periods.

Short Comparison with CBT

The biggest difference to CBT is that time slots have gained a new dimension. This enables efficient scheduling of aperiodic courses but expands the search space massively.

In CBT, the set of courses a student has to attend is called a curriculum. The courses contained in a curriculum must be scheduled conflict-free, so that every student can attend all classes without conflict. In *Aperiodic Curriculum-Based Timetabling (Aperiodic CBT)*, however, curricular constraints are modeled by groups. For every base group all classes have to be scheduled conflict-free.

Recall from [Chapter 1](#) that teachers are not considered explicitly in Aperiodic CBT. Instead a set of eligible combinations of organizations is given per class. Then, per class a combination of organizations needs to be matched. This makes the search space for the decision problem bigger compared to CBT. We will take care of this shortly in [Section 2.2](#).

Whereas in CBT a conflict-free schedule per teacher is required, we only consider the capacities of organizations here. These give upper bounds on the amount of teaching an organization can deliver in parallel, in a day and in a week.

2.1 Notation

Subsequently, we list the notation used in this work. To denote the power set of a set S we will write 2^S . Also we will use $[k]$ as shorthand for $\{i \in \mathbb{N} \mid 1 \leq i \leq k\}$ and $\exists! x \varphi$ meaning that there exists exactly one x so that φ evaluates true. Integer linear programming will be abbreviated by ILP.

Resources We will refer to the following sets of resources:

\mathcal{C}	a set of classes
$\hat{\mathcal{C}}$	a set of courses, where each course is a set of classes and each class belongs to exactly one course
$\hat{c}: \mathcal{C} \rightarrow \hat{\mathcal{C}}$	a function that gives the exact course a class belongs to
\mathcal{R}	a set of rooms
\mathcal{O}	a set of organizations
$\hat{\mathcal{O}} \subseteq 2^{\mathcal{O}}$	a set containing sets of organizations, each being called a combination of organizations

\mathcal{B} a set of base groups
 $\mathcal{G} \subseteq 2^{\mathcal{B}}$ a set of groups, where each group is a set of base groups

Time We will refer to the following sets, representing time:

$\mathcal{W} \subseteq \mathbb{N}$ weeks of a semester
 $\mathcal{D} \subseteq \mathbb{N}$ days of a week
 $\mathcal{H} \subseteq \mathbb{N}$ pairwise disjoint time intervals
 $\mathcal{T} \subseteq \mathcal{W} \times \mathcal{D} \times \mathcal{H}$ time slots
 $\mathcal{T}(w) \subseteq \mathcal{T}$ time slots in week w
 $\mathcal{T}(d) \subseteq \mathcal{T}$ time slots where day of week is d (e.g. all time slots on Mondays)
 $\mathcal{P} \subseteq 2^{\mathcal{T}}$ periods
 $\mathcal{P}(w) \subseteq \mathcal{P}$ periods in week w
 $\mathcal{P}(d) \subseteq \mathcal{P}$ periods where day of week is d (e.g. all periods on Mondays)
 $\mathcal{P}(t) \subseteq \mathcal{P}$ periods containing time slot t

If a period contains more than one set of consecutive time slots, then we will refer to each consecutive set as a session. The following functions provide further information about time slots and periods:

$\text{length}_t: \mathcal{T} \rightarrow \mathbb{N}$ length of a time slot in minutes

$\text{length}_p: \mathcal{P} \rightarrow \mathbb{N}$ length of a period in minutes

Course Properties The properties of a given course are described by the following functions:

$\mathcal{G}_{\hat{c}}: \hat{\mathcal{C}} \rightarrow 2^{\mathcal{G}}$ gives a set of groups that have to attend the course

$\hat{\mathcal{O}}_{\hat{c}}: \hat{\mathcal{C}} \rightarrow 2^{\hat{\mathcal{O}}}$ gives a set of combinations of organizations eligible for the course

$\mathcal{C}_{\text{count}}: \hat{\mathcal{C}} \times \hat{\mathcal{O}} \rightarrow \mathbb{N}$ gives the number of classes in a course to be taught by a combination of organizations

$\mathcal{P}_{\hat{c},g}: \hat{\mathcal{C}} \times \mathcal{G} \rightarrow 2^{\mathcal{P}}$ gives a set of eligible periods for a course per group

$\mathcal{R}_{\hat{c},\hat{o}}: \hat{\mathcal{C}} \times \hat{\mathcal{O}} \rightarrow 2^{\mathcal{R}}$ gives a set of eligible rooms for a course per combination of organizations

Availabilities Rooms and organizations can be blocked for certain times. The following functions provide restrictions on the available times:

$\mathcal{P}_r: \mathcal{R} \rightarrow 2^{\mathcal{P}}$ gives the set of available periods for a room

$\mathcal{P}_o: \mathcal{O} \rightarrow 2^{\mathcal{P}}$ gives the set of available periods for an organization

$\mathcal{P}_{\hat{o}}: \hat{\mathcal{O}} \rightarrow 2^{\mathcal{P}}$ gives the set of available periods for a combination of organizations, being the intersection of the availabilities of the contained organizations

Capacities The capacity functions give us upper bounds on the teaching an organization can perform in a given time frame. They are defined as following:

$\text{cap}_{\text{parallel}}: \mathcal{O} \times \mathcal{T} \rightarrow \mathbb{N}$ gives the maximum number of classes an organization can teach at a given time slot

$\text{cap}_{\text{daily}}: \mathcal{O} \times \mathcal{W} \times \mathcal{D} \rightarrow \mathbb{N}$ gives the maximum minutes an organization can teach on a given date

$\text{cap}_{\text{weekly}}: \mathcal{O} \times \mathcal{W} \rightarrow \mathbb{N}$ gives the maximum minutes an organization can teach in a given week

Preferences Organizations like to state which times they prefer to teach in. Also per course, each eligible combination of organizations can state which of the eligible rooms they prefer. The following functions provide these measures. The lower the number, the higher the priority.

$\text{pref}_{o,t}: \mathcal{O} \times \mathcal{T} \rightarrow \mathbb{N}$ gives an organization's preference regarding a given time slot

$\text{pref}_{\hat{c},r,\hat{o}}: \hat{\mathcal{C}} \times \mathcal{R} \times \hat{\mathcal{O}} \rightarrow \mathbb{N}$ gives the preference of a combination of organizations regarding a given room for a given course

2.2 Assigning Organizations to Classes

Recall that for each course, a set of eligible combinations of organizations is given, along with how many classes each combination should teach. It is ensured that in sum these counts match the number of classes. To shrink the search space of the decision problem, we compute a matching between classes and combinations of organizations respecting the eligibility and counts as illustrated in [Figure 2.4](#).

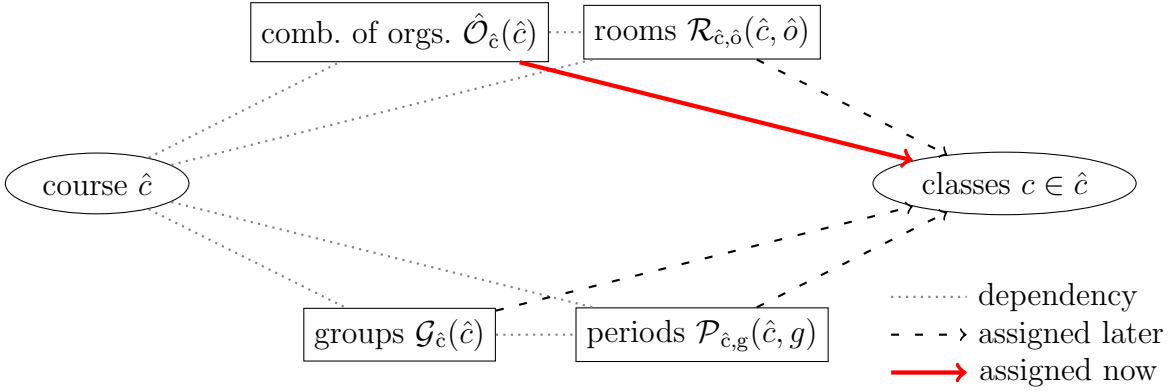


Figure 2.4: Assigning a combination of organizations to each class. Highlighted as a solid red arrow is the assignment of a combination of organizations to a class performed in this step. For a more detailed description refer to [Figure 2.3](#).

The following algorithm will assign a combination of organizations to each class:

Algorithm 1: Assigning Combinations of Organizations to Classes

Input:

- A set of courses $\hat{\mathcal{C}}$
- A function $\hat{\mathcal{O}}_{\hat{\mathcal{C}}}: \hat{\mathcal{C}} \rightarrow 2^{\hat{\mathcal{O}}}$ that defines the set of eligible combinations of organizations per course
- A function $\mathcal{C}_{\text{count}}: \hat{\mathcal{C}} \times \hat{\mathcal{O}} \rightarrow \mathbb{N}$ that states how many classes of a course an eligible combination of organizations should teach

Output:

- A function $\hat{o}: \mathcal{C} \rightarrow \hat{\mathcal{O}}$ that maps every class to a combination of organizations

for $\hat{c} = \{c_1, \dots, c_k\} \in \hat{\mathcal{C}}$ **do**

$i := 1$

for $\hat{o} \in \hat{\mathcal{O}}_{\hat{c}}$ **do**

for $j := 0; j < \mathcal{C}_{\text{count}}(\hat{c}, \hat{o}); j := j + 1$ **do**

$l := i + j$

$\hat{o}(c_l) := \hat{o}$

$i := i + \mathcal{C}_{\text{count}}(\hat{c}, \hat{o})$

The outer loop iterates over all courses, whereas the inner two loops together iterate over the classes of the course by first iterating over the eligible combinations of organizations for the course and then counting the number of classes that need to be assigned as per $\mathcal{C}_{\text{count}}$. None of the used sets and relations change during the execution other than the output function \hat{o} . Also, there cannot be more eligible combinations of organizations per course than classes. The function $\mathcal{C}_{\text{count}}$ is guaranteed to sum up to the exact number of classes per course. Hence, [Algorithm 1](#) runs in linear time only depending on the number of classes.

From now on we will a priori assume that every class has had its combination of organizations matched.

Remark. Precomputing this matching does not impair the feasibility of the decision problem since no assignable property has been attached to a class at this point. Therefore, no generality is lost.

To simplify further definitions we define a function $\mathcal{C}_o: \mathcal{O} \rightarrow 2^{\mathcal{C}}$ as follows:

$$\mathcal{C}_o(o) \mapsto \{c \in \mathcal{C} \mid o \in \hat{o}(c)\}.$$

2.3 Definition of the Problem

Now that we have assigned a combination of organizations to each class, we can define the set of all eligible assignments.

Definition 2.1. Let \mathcal{A} to be the set of all eligible assignments, also called the *matching-space*:

$$\begin{aligned} \mathcal{A} := \{ & (c, t, r, b) \mid c \in \mathcal{C} \\ & \wedge \exists p \in \mathcal{P}_{\hat{c},g}(\hat{c}(c), g) \cap \mathcal{P}_r(r) \cap \mathcal{P}_{\hat{o}}(\hat{o}(c)): t \in p \\ & \wedge r \in \mathcal{R}_{\hat{c},\hat{o}}(\hat{c}(c), \hat{o}(c)) \\ & \wedge \exists g \in \mathcal{G}_{\hat{c}}(\hat{c}(c)): b \in g \} \end{aligned}$$

This matching-space contains all eligible combinations of classes, their time slots, rooms and base groups as tuples. A subset of \mathcal{A} is a valid assignment of a period, room and group to a class, if the time slots within the subset are an eligible period and the base groups within the subset are an eligible group. A valid assignment for a class c_1 could be

$$\{(c_1, t_{1,1}, r_1, b_{1,1}), (c_1, t_{1,2}, r_1, b_{1,1}), (c_1, t_{1,1}, r_1, b_{1,2}), (c_1, t_{1,2}, r_1, b_{1,2})\} \subseteq \mathcal{A}$$

if an eligible period $p_1 = \{t_{1,1}, t_{1,2}\}$ and group $g_1 = \{b_{1,1}, b_{1,2}\}$ existed.

2.3.1 Conflicts

Having defined the matching-space \mathcal{A} , it is easy to see that not all assignments are compatible with each other. For example, a room must not be booked by two classes simultaneously. To avoid conflicts in general we will define a conflict graph H_{conf} , which is a hypergraph. Its vertices are all eligible assignments \mathcal{A} and edges represent conflicts between assignments. For every edge, all except one contained assignments are valid in combination. There are several parts to the conflict graph.

Groups No base group can attend two classes in parallel. So for all intersecting groups, all combinations of intersecting periods are to be avoided. Hence, we define a set of two-element edges

$$E_{\mathcal{G}} := \{\{a_1 = (c_1, t, r_1, b), a_2 = (c_2, t, r_2, b)\} \mid a_1, a_2 \in \mathcal{A} \wedge c_1 \neq c_2\}.$$

Rooms No room can host two classes in parallel. So for every room every combination of conflicting periods has to be forbidden. Again we define a set of two-element edges

$$E_{\mathcal{R}} := \{\{a_1 = (c_1, t, r, b_1), a_2 = (c_2, t, r, b_2)\} \mid a_1, a_2 \in \mathcal{A} \wedge c_1 \neq c_2\}.$$

Capacities To keep the load of organizations below their stated capacities, we define three sets of edges, one for every time frame that capacities are given in. They include all combinations of possible assignments that together would exceed the capacities.

For every combination of assignments that would overload a given organization and week, we create an edge.

$$\begin{aligned} E_{\text{cap}_{\text{weekly}}} := & \{\{a_1 = (c_1, t_1, r_1, b_1), \dots, a_k = (c_k, t_k, r_k, b_k)\} \mid a_1, \dots, a_k \in \mathcal{A} \\ & \wedge \forall i, j \in [k]^2 \ i \neq j: (t_i \neq t_j) \vee (c_i \neq c_j) \\ & \wedge \exists o \in \mathcal{O} \ \forall i \in [k]: o \in \hat{o}(c_i) \\ & \wedge \exists w \in \mathcal{W} \ \forall i \in [k]: t_i \in \mathcal{T}(w) \\ & \wedge \sum_{i \in [k]} \text{length}_t(t_i) > \text{cap}_{\text{weekly}}(o, w)\} \end{aligned}$$

We do the same for daily capacities on each given combination of week and day of week.

$$\begin{aligned} E_{\text{cap}_{\text{daily}}} := & \{\{a_1 = (c_1, t_1, r_1, b_1), \dots, a_k = (c_k, t_k, r_k, b_k)\} \mid a_1, \dots, a_k \in \mathcal{A} \\ & \wedge \forall i, j \in [k]^2 \ i \neq j: (t_i \neq t_j) \vee (c_i \neq c_j) \\ & \wedge \exists o \in \mathcal{O} \ \forall i \in [k]: o \in \hat{o}(c_i) \\ & \wedge \exists w \in \mathcal{W} \ \exists d \in \mathcal{D} \ \forall i \in [k]: t_i \in \mathcal{T}(w) \cap \mathcal{T}(d) \\ & \wedge \sum_{i \in [k]} \text{length}_t(t_i) > \text{cap}_{\text{daily}}(o, w, d)\} \end{aligned}$$

For parallel capacities, we count classes instead of minutes. Consequently, we create edges for every combination of assignments that would exceed the given maximum number of classes per time slot for every organization.

$$\begin{aligned} E_{\text{cap}_{\text{parallel}}} := & \{\{a_1 = (c_1, t, r_1, b_1), \dots, a_k = (c_k, t, r_k, b_k)\} \mid a_1, \dots, a_k \in \mathcal{A} \\ & \wedge \forall i, j \in [k]^2 \ i \neq j: (c_i \neq c_j) \\ & \wedge \exists o \in \mathcal{O} \ \forall i \in [k]: o \in \hat{o}(c_i) \\ & \wedge k = \text{cap}_{\text{parallel}}(o, t) + 1\} \end{aligned}$$

Finally, we can define the conflict graph.

Definition 2.2. H_{conf} denotes the conflict graph, being a hypergraph with

$$\begin{aligned} V(H_{\text{conf}}) &:= \mathcal{A}, \\ E(H_{\text{conf}}) &:= E_{\mathcal{G}} \cup E_{\mathcal{R}} \cup E_{\text{capweekly}} \cup E_{\text{capdaily}} \cup E_{\text{capparallel}} \subseteq 2^{\mathcal{A}}. \end{aligned}$$

Now we will define the three constraints a feasible schedule has to obey.

Definition 2.3. We call $\mathcal{L} \subseteq \mathcal{A}$ a feasible schedule if

$$\forall c \in \mathcal{C} \exists! p \in \mathcal{P} \forall t \in p \exists! r \in \mathcal{R} \exists! g \in \mathcal{G} \forall b \in g: (c, t, r, b) \in \mathcal{L} \quad (\text{taking place (1)})$$

$$\forall \hat{c} \in \hat{\mathcal{C}} \forall g \in \mathcal{G}_{\hat{c}}(\hat{c}) \forall b \in g \exists! c \in \hat{c} \exists p \in \mathcal{P} \forall t \in p \exists r \in \mathcal{R}: (c, t, r, b) \in \mathcal{L} \quad (\text{group obligation (2)})$$

$$\forall e \in E(H_{\text{conf}}): \mathcal{L} \cap e \neq e \quad (\text{conflicts (3)})$$

Then the formalization of the decision problem is:

APERIODIC CURRICULUM-BASED TIMETABLING

Input: The set \mathcal{A} of all possible assignments and the hypergraph H_{conf} .

Task: Find a feasible schedule $\mathcal{L} \subseteq \mathcal{A}$.

2.4 Key Observations

For any nontrivial instance, the conflict graph H_{conf} will become rather large. Consider an instance with roughly the following dimensions:

- $|\mathcal{C}| = 15,000$,
- $|\mathcal{T}| = 1,000$,
- $|\mathcal{R}| = 200$,
- $|\mathcal{B}| = 1,100$,

hence the product being $3.3 \cdot 10^{12}$. Holding a graph of this size in memory would require at least 13.2 Tebibyte for the vertices alone, assuming a 32-Bit system and only counting references. Thus, we will determine how to cope with its size in the following.

For most data reduction rules on hypergraphs upper bounds on the edges are needed (see i.e. [Abu10]). We will derive some properties of the edges of our hypergraph H_{conf} here.

2.4.1 Size of the Conflict Graph

By definition, all edges contained in $E_{\mathcal{G}}$ and $E_{\mathcal{R}}$ are of size two. Their endpoints also have the same time slot. Regarding time slots, the same applies to all edges in $E_{\text{capparallel}}$ but

$$\max_{e \in E_{\text{capparallel}}} |e| = \max_{o \in \mathcal{O}} \max_{t \in \mathcal{T}} \text{cap}_{\text{parallel}}(o, t) + 1.$$

So the maximum sizes of edges depend solely on the capacities specified by the planner. In the worst case, the maximum number of parallel classes would be limited by the number of classes held by an organization altogether since no more than all classes can be scheduled in parallel.

In contrast, the edges in $E_{\text{cap}_{\text{weekly}}}$ and $E_{\text{cap}_{\text{daily}}}$ span multiple time slots. Counting the number of classes that could be held in a given week or day is nontrivial since their eligible periods can contain combinations of time slots that span multiple days or even weeks. So we focus on a more easily computable upper bound. Since once again capacities are given per organization and we know which class is held by which combination of organizations, we count the classes per organization and multiply with the cardinality of the domains for the other resources part of our tuples. We obtain

$$\max_{e \in E_{\text{cap}_{\text{weekly}}}} |e| = \max_{o \in \mathcal{O}} \max_{w \in \mathcal{W}} |\mathcal{C}_o(o)| \cdot |\mathcal{T}(w)| \cdot |\mathcal{R}| \cdot |\mathcal{B}|$$

and

$$\max_{e \in E_{\text{cap}_{\text{daily}}}} |e| = \max_{o \in \mathcal{O}} \max_{w \in \mathcal{W}} \max_{d \in \mathcal{D}} |\mathcal{C}_o(o)| \cdot |\mathcal{T}(w) \cap \mathcal{T}(d)| \cdot |\mathcal{R}| \cdot |\mathcal{B}|.$$

Since the edges contain all combinations of tuples that cause more load than the stated capacities, there exists at least one edge per class that contains all tuples concerned with eligible time slots in the given week and date, respectively. Unfortunately, there are no meaningful restrictions possible for the number of rooms and base groups per organization. Every organization could be assigned every group of a course it participates in and the same goes for rooms.

This results in no usable upper bounds for the size of edges in the conflict graph. But as seen in the above-described analysis of the edges of H_{conf} , there is no edge that exceeds the time slots of one week. We can use this to our advantage and partition the whole graph into weeks. This gives us up to 16 separate and independent conflict graphs since a typical semester consists of roughly that many weeks.

This partitioning can only be applied to the conflict graph, however, so we cannot solve 16 independent instances of APERIODIC CBT. This is because even a single period can tie the whole semester together because it has sessions in every week. Thus all conflicts stay global problems. The only reason the partitioning of the conflict graph works is because the vertices only contain time slots, which are all disjoint in contrast to periods.

2.4.2 Connection to Hitting Set

In order not to violate the *conflicts* (3) constraint we must not take more than $|e| - 1$ elements per edge e . This translates properly to HITTING SET, where of each edge at least one element has to be chosen. We can use a hitting set of our conflict graph to eliminate all conflicts by deleting the chosen elements from the matching-space \mathcal{A} . No edge of the conflict graph can then be part of the solution in its entirety.

HITTING SET

Input: A hypergraph H with vertices V_H and edges E_H .

Task: Find a hitting set $S \subseteq V_H$ so that $\forall e \in E_H \exists s \in S: s \in e$ holds.

2 Model & Key Observations

Formally, when a hitting set $\mathcal{L}_{HS} \subseteq \mathcal{A}$ was found, we try to find a solution $\mathcal{L} \subseteq \mathcal{A} \setminus \mathcal{L}_{HS}$. This time, we only need to take care of the two remaining constraints *taking place (1)* and *group obligation (2)*. Still we have no guarantee that such a solution exists solely because a hitting set exists. What can be done is constructing a hitting set from a solution to APERIODIC CBT.

Lemma 2.4. *Let \mathcal{I} be a yes-instance with solution \mathcal{L} for APERIODIC CBT. Then $\mathcal{A} \setminus \mathcal{L}$ is a hitting set for H_{conf} .*

Proof. Directly following constraint *conflicts (3)*, we see that there exists a non-empty part $A' \subseteq \mathcal{A}$ of every edge e in H_{conf} that must not be part of \mathcal{L} . So we construct a hitting set out of these parts which is exactly $\mathcal{A} \setminus \mathcal{L}$. \square

However, to find a hitting set on a hypergraph is NP-complete. Therefore we need to employ heuristics or data reduction rules to find a solution quickly. But as seen in the previous subsection, computing useful upper bounds on the size of the edges is not easily done.

Also, we may find a hitting set that, when incorporated, leaves an infeasible problem. However, as seen in the previous proof, we can always extract a hitting set solution from any feasible solution to our problem.

2.4.3 A Simple Reduction Rule

In the following reduction rule, we recognize a fixation in the input data. When a course only has one eligible period, room and group, there is no use in leaving it in the decision problem. Instead we take the classes out of the input and block the room and group for the scheduled time. Also, we lower the capacities of the scheduled organizations accordingly.

Note that we do not have to take care of organizational capacities here. If capacities were too low to schedule this course, then the problem would be infeasible to begin with.

We define the necessary preconditions and modifications for the described rule.

Reduction Rule 2.4.1. *Let there be a course $\hat{c}^* \in \hat{\mathcal{C}}$ so that*

$$\begin{aligned} \hat{c}^* &= \{c^*\}, \\ \hat{\mathcal{O}}_{\hat{c}}(\hat{c}^*) &= \{\hat{o}^*\}, \\ \mathcal{G}_{\hat{c}}(\hat{c}^*) &= \{g^*\}, \\ \mathcal{R}_{\hat{c}, \hat{o}}(\hat{c}^*, \hat{o}^*) &= \{r^*\}, \\ \mathcal{P}_{\hat{c}, g}(\hat{c}^*, g^*) \cap \mathcal{P}_r(r^*) \cap \mathcal{P}_{\hat{o}}(\hat{o}^*) &= \{p^*\}, \\ \mathcal{C}_{\text{count}}(\hat{c}^*, \hat{o}^*) &= 1. \end{aligned}$$

Then we can take this course entirely out of our matching-space. We perform the following modifications on our input:

$$\begin{aligned}\mathcal{C}' &:= \mathcal{C} \setminus \{c^*\}, \\ \hat{\mathcal{C}}' &:= \hat{\mathcal{C}} \setminus \{\hat{c}^*\}, \\ \mathcal{P}'_r(r^*) &:= \mathcal{P}_r(r^*) \setminus \{p \in \mathcal{P} \mid p \cap p^* \neq \emptyset\} \\ \mathcal{P}'_{\hat{c},g}(\hat{c}, g) &:= \mathcal{P}_{\hat{c},g}(\hat{c}, g) \setminus \{p \in \mathcal{P} \mid p \cap p^* \neq \emptyset\} \quad \forall \hat{c} \in \hat{\mathcal{C}} \forall g \in \{g \in \mathcal{G} \mid g \cap g^* \neq \emptyset\}, \\ \mathcal{G}'_{\hat{c}}(\hat{c}^*) &:= \emptyset.\end{aligned}$$

This shrinks the matching-space accordingly:

$$\begin{aligned}\mathcal{A}' &:= \{(c, t, r, b) \mid c \in \mathcal{C}' \\ &\quad \wedge \exists p \in \mathcal{P}'_{\hat{c},g}(\hat{c}(c), g) \cap \mathcal{P}'_r(r) \cap \mathcal{P}_{\hat{o}}(\hat{o}(c)): t \in p \\ &\quad \wedge r \in \mathcal{R}_{\hat{c},\hat{o}}(\hat{c}(c), \hat{o}(c)) \\ &\quad \wedge \exists g \in \mathcal{G}_{\hat{c}}(\hat{c}(c)): b \in g\}.\end{aligned}$$

Also, the conflict graph is restricted to the new matching-space

$$\begin{aligned}V(\mathbb{H}'_{\text{conf}}) &:= \mathcal{A}' \\ E(\mathbb{H}'_{\text{conf}}) &:= \{e' \mid e \in E(\mathbb{H}_{\text{conf}}) \wedge e' = e \cap \mathcal{A}'\}.\end{aligned}$$

Lemma 2.5. *Let \mathcal{I} be an instance of APERIODIC CBT with solution \mathcal{L} and let \mathcal{I}' be \mathcal{I} after applying the [Reduction Rule 2.4.1](#) with solution \mathcal{L}' . Then \mathcal{I}' is a yes-instance if and only if \mathcal{I} is a yes-instance.*

Proof. Let \mathcal{L}^* be the set of assignments that are fixed by the reduction rule, that is,

$$\mathcal{L}^* := \bigcup_{t_i \in p^*} \bigcup_{b_j \in g^*} \{(c^*, t_i, r^*, b_j)\}.$$

(\Rightarrow) First, we show that \mathcal{L}' is a solution for \mathcal{I}' when $\mathcal{L}' \cup \mathcal{L}^*$ is a solution for \mathcal{I} . We observe that every class $c \in \mathcal{C}$ in \mathcal{I} fulfills the *taking place (1)* constraint. This holds since all necessary assignments are either in \mathcal{L}' if $c \neq c^*$ or in \mathcal{L}^* if $c = c^*$.

Second, every group $g \in \mathcal{G}$ in \mathcal{I} fulfills the *group obligation (2)* constraint. As per precondition only group g^* needs to attend the one class c^* of course \hat{c}^* . These necessary assignments are all in \mathcal{L}^* . So for every class $c \in \mathcal{C}$, all necessary assignments are either in \mathcal{L}' if $c \neq c^*$ or in \mathcal{L}^* if $c = c^*$, as per construction of \mathcal{L}^* .

Lastly, we note that every conflict encoded in $e \in E(\mathbb{H}_{\text{conf}})$ fulfills the *conflicts (3)* constraint. Therefore, assume towards a contradiction that there is a set $A \subseteq \mathcal{L}^*$, an edge $e' \in E(\mathbb{H}'_{\text{conf}})$, and an edge $e \in E(\mathbb{H}_{\text{conf}})$ with $e' \cup A = e$ and $e \cap (\mathcal{L}' \cup \mathcal{L}^*) = e'$. Then we have a no-instance of \mathcal{I} . We observe that $e' \cap \mathcal{L}' = e'$ because $e' \subset 2^{\mathcal{A}'}$. This contradicts \mathcal{L}' being a solution of \mathcal{I}' .

2 Model & Key Observations

(\Leftarrow) Now we show that \mathcal{L} is a solution for \mathcal{I} when $\mathcal{L}' = \mathcal{L} \setminus \mathcal{L}^*$ is a solution for \mathcal{I}' . Firstly, for every class $c \in \mathcal{C}'$ in \mathcal{I} , the *taking place (1)* constraint holds because for every class $c \in \mathcal{C}$, all necessary assignments are either in \mathcal{L}' if $c \neq c^*$ or in \mathcal{L}^* if $c = c^*$. From $\mathcal{C}' = \mathcal{C} \setminus \{c^*\}$ follows that the *taking place (1)* constraint is fulfilled.

Every group $g \in \mathcal{G}'$ in \mathcal{I} fulfills the *group obligation (2)* constraint. This holds because $\mathcal{G}'_{\hat{c}}(c^*) = \emptyset$, as per construction of $\mathcal{G}'_{\hat{c}}$. As all assignments for c^* are in \mathcal{L}^* , none are in $\mathcal{L}' = \mathcal{L} \setminus \mathcal{L}^*$.

Lastly, every conflict encoded in $e \in E(H'_{\text{conf}})$ fulfills the *conflicts (3)* constraint. We observe that every edge $e' \in E(H'_{\text{conf}})$ exists either unaltered as $e \in E(H_{\text{conf}})$ or there is $A \subseteq \mathcal{A}$ so that $e' \cup A = e$. Then $A \subseteq \mathcal{A} \setminus \mathcal{A}'$ and therefore $\mathcal{L}' \cap e' \neq e'$ holds for all $e' \in E(H'_{\text{conf}})$.

This concludes the proof. □

3 Proposed Heuristic

In this chapter, we tackle APERIODIC CURRICULUM-BASED TIMETABLING (APERIODIC CBT) using integer programming. This has been a common approach for timetabling problems [LL12; Sch99]. Moreover, a lot of effort is put into the continuous development of several commercial integer programming solvers.

Though solving CURRICULUM-BASED TIMETABLING (CBT) using integer programming is well studied [Bon+12; McC+10] the differences already mentioned in Chapter 2 expand the search space for APERIODIC CBT greatly compared to CBT. Therefore, to get a feasible timetable within acceptable time, the problem will be solved heuristically in two steps. Before applying an ILP, we already assigned combinations of organizations to classes in Section 2.2. Now, in Step 1, we will assign groups to classes, before a period and room are assigned in step 2. Together, both steps will produce a feasible schedule as defined by APERIODIC CBT.

In the end, we will discuss a real world implementation and its performance. Further, an additional step between 1 and 2 will be described, which reduces the search space.

3.1 Step 1

To reduce the size of our *matching space* (see Definition 2.1), we will assign groups to classes (see Figure 3.1) independently of period and room assignments in a first step. For every course, there are exactly as many classes as groups that have to attend. Therefore, per course, a bijection from classes to groups can be computed.

Since the set of eligible periods for a class depends heavily on the assigned group, we have to take capacities into consideration. Recall that we already assigned an eligible combination of organizations to each class in Section 2.2. Therefore, we must not assign too many groups with overlapping sets of eligible periods to the same organization. The goal is to keep the average loads below the capacities. Otherwise, the *group assignment* could result in an infeasible next step.

To get an approximate measure on the load of assigning a specific group, we define a function $\text{avg}_{\mathcal{P}, \mathcal{T}}$ that provides the average share of minutes a set of periods has within a given set of time slots:

$$\text{avg}_{\mathcal{P}, \mathcal{T}}: 2^{\mathcal{P}} \times 2^{\mathcal{T}} \rightarrow \mathbb{Q}.$$

We will apply this function on the set of eligible periods $\mathcal{P}_{\hat{c}, g}(\hat{c}, g)$ for some course \hat{c} and group g and a set of time slots concerned with a given capacity constraint, i.e., all time slots $\mathcal{T}(w)$ of some week w . The result will be the average length of each period within the given set of time slots. To calculate the length of a period p within a set of time slots T , the lengths of the time slots in the intersection $p \cap T$ are summed up.

3 Proposed Heuristic

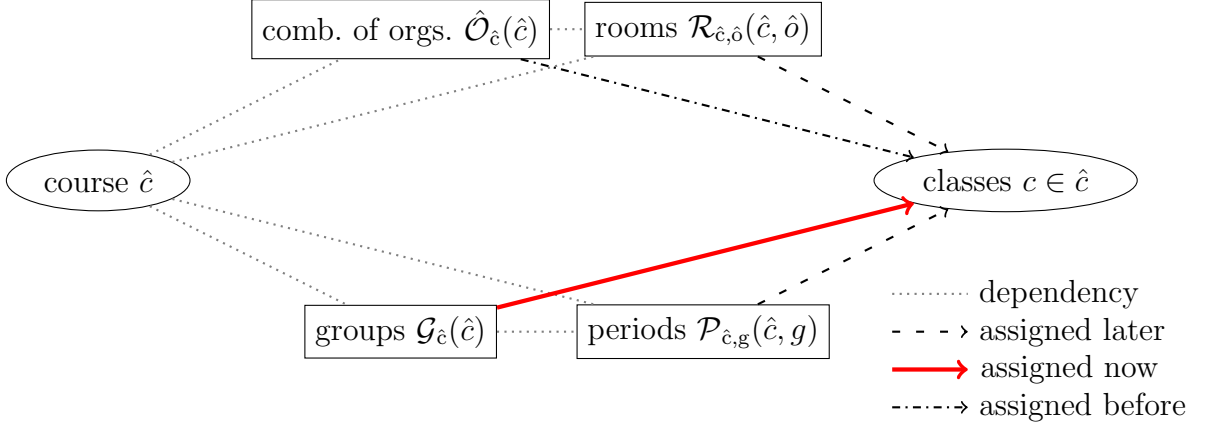


Figure 3.1: Assigning a group to each class. Highlighted as a solid red arrow is the assignment of a group to a class, performed in this step. The combinations of organizations have already been assigned in Section 2.2. For a more detailed description refer to Figure 2.3.

Now we will define the constraints a feasible *group assignment* has to obey. Each class needs to be assigned to a group. Also, every group that has to attend a course has to be assigned to one class of the course. These two conditions result in one constraint, which we copy directly from constraint *group obligation (2)* of Section 2.3. Other than that, we need two constraints for weekly and daily capacities. Each one sums up the average lengths of classes assigned to a specific organization per time frame concerning a capacity and keeps this sum below the stated maximum.

We will refer to the result via the function $g: \mathcal{C} \rightarrow \mathcal{G}$ that maps each class to a group.

Definition 3.1. We call the function g a feasible *group assignment* if

$$\forall \hat{c} \in \hat{\mathcal{C}} \quad \forall g \in \mathcal{G}_{\hat{c}}(\hat{c}) \quad \exists ! c \in \hat{c}: g(c) = g$$

(group obligation (1))

$$\forall o \in \mathcal{O} \quad \forall w \in \mathcal{W}: \sum_{c \in \mathcal{C}_o(o)} \text{avg}_{\mathcal{P}, \mathcal{T}}(\mathcal{P}_{\hat{c}, g}(\hat{c}(c), g(c)), \mathcal{T}(w)) \leq \text{cap}_{\text{weekly}}(o, w)$$

(weekly capacity (2))

$$\forall o \in \mathcal{O} \quad \forall w \in \mathcal{W} \quad \forall d \in \mathcal{D}: \sum_{c \in \mathcal{C}_o(o)} \text{avg}_{\mathcal{P}, \mathcal{T}}(\mathcal{P}_{\hat{c}, g}(\hat{c}(c), g(c)), \mathcal{T}(w) \cap \mathcal{T}(d)) \leq \text{cap}_{\text{daily}}(o, w, d).$$

(daily capacity (3))

Then the formalization of the decision problem is:

APERIODIC CURRICULUM-BASED TIMETABLING - STEP 1

Input: The set \mathcal{A} of all possible assignments and the function $\hat{o}: \mathcal{C} \rightarrow \hat{\mathcal{O}}$ from Section 2.2.

Task: Find a feasible group assignment g .

3.1.1 NP-completeness

Obviously, the correctness of a group assignment can be checked in polynomial time. Therefore, the problem lies in NP. To prove that APERIODIC CBT - STEP 1 is also NP-complete, we will give a polynomial-time reduction from 3-SAT.

3-SAT

Input: A boolean formula in conjunctive normal form where each clause has exactly three different literals (3-CNF).

Task: Find an assignment that satisfies the given formula.

Theorem 3.2. APERIODIC CURRICULUM-BASED TIMETABLING - STEP 1 *is NP-hard.*

Proof. Let φ be a boolean formula in 3-CNF with m clauses and n variables in the form of:

$$\varphi = \bigwedge_{i=1}^m K_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}),$$

where K_i are clauses and $\ell_{i,j}$ are literals. We refer to the set of variables by $\text{var}(\varphi)$ and the set of clauses by $\text{clause}(\varphi)$. We construct an instance of APERIODIC CBT - STEP 1 in the following.

Construction We use four weeks $\{w_1, \dots, w_4\}$ to schedule the classes into. Each week has a single day of week d_1 . The general idea is to have the scheduling of week one correlate with the variable assignment. Other weeks exist solely to elude week one when its capacity is reached. We will introduce courses for variables and clauses. Classes that conflict based on their variable usage will be linked to the same organization. The capacities will take care of not being able to schedule conflicting assignments. Of each course, representing a clause exactly one class representing a literal will have to be scheduled in week one. This literal will be the one satisfying its clause.

We will now provide a more detailed construction. The instance of APERIODIC CBT will have the course set $\hat{\mathcal{C}} := \hat{\mathcal{C}}_x \cup \hat{\mathcal{C}}_K$ for variables and clauses. Analogously, the same goes for the set of groups $\mathcal{G} := \mathcal{G}_x \cup \mathcal{G}_K$. The above sets will be defined in the following paragraphs.

Variables For every variable $x_i \in \text{var}(\varphi)$ we introduce a so-called variable-course \hat{c}_{x_i} . The course contains exactly two classes, one for every choice of assigning true or false. Also, two combinations of organizations are introduced per variable, again for representing true and false assignments. Each combination of organizations consists of a single unique organization. Formally:

$$\begin{aligned} \hat{\mathcal{C}}_x &:= \{\hat{c}_{x_i} = \{c_{x_i}, c_{\bar{x}_i}\} \mid x_i \in \text{var}(\varphi)\} \\ \hat{\mathcal{O}} &:= \{\hat{o}_{x_i} = \{o_{x_i}\}, \hat{o}_{\bar{x}_i} = \{o_{\bar{x}_i}\} \mid x_i \in \text{var}(\varphi)\}. \end{aligned}$$

3 Proposed Heuristic

Normally, the function $\hat{\delta}: \mathcal{C} \rightarrow \hat{\mathcal{O}}$ is precomputed using [Algorithm 1](#) of [Section 2.2](#). Here, we use it to link the variable-classes to their corresponding organizations as follows:

$$\begin{aligned}\hat{\delta}(c_{x_i}) &:= \hat{\delta}_{x_i} & \forall x_i \in \text{var}(\varphi) \\ \hat{\delta}(c_{\bar{x}_i}) &:= \hat{\delta}_{\bar{x}_i} & \forall x_i \in \text{var}(\varphi)\end{aligned}$$

Two groups $\{g_{x_i}^1, g_{x_i}^4\}$ are introduced which need to be assigned to classes c_{x_i} and $c_{\bar{x}_i}$. Which group is assigned to which variable-class will decide the assignment for the corresponding variable. The upper indices of the groups will signify the eligible week. If group $g_{x_i}^1$ is assigned to class c_{x_i} , the class will be scheduled in week one. Therefore, x_i will be set to true. The groups all consist of a single unique base group and are linked to their corresponding course as follows:

$$\begin{aligned}\mathcal{G}_x &:= \{g_{x_i}^1, g_{x_i}^4 \mid x_i \in \text{var}(\varphi)\} \\ \mathcal{G}_{\hat{c}}(\hat{c}_{x_i}) &:= \{g_{x_i}^1, g_{x_i}^4\} & \forall x_i \in \text{var}(\varphi)\end{aligned}$$

As mentioned, we set the eligible periods for all groups according to their upper index, which will indicate the eligible week. Group $g_{x_i}^1$, if assigned to a class, will force the class of course \hat{c}_{x_i} into week one, whereas group $g_{x_i}^4$ will force the class of course \hat{c}_{x_i} into week four. All eligible periods for variable-courses and their eligible groups will have length m , which is the number of clauses. This is achieved by choosing a time interval with the appropriate length.

$$\begin{aligned}\mathcal{P}_{\hat{c},g}(\hat{c}_{x_i}, g_{x_i}^1) &:= \{p_x^1 = (w_1, d_1, [m])\} & \forall x_i \in \text{var}(\varphi) \\ \mathcal{P}_{\hat{c},g}(\hat{c}_{x_i}, g_{x_i}^4) &:= \{p_x^4 = (w_4, d_1, [m])\} & \forall x_i \in \text{var}(\varphi)\end{aligned}$$

To summarize the construction for variables, we show the two possible schedules for a variable x_i in [Figure 3.2](#).

Clauses & Literals After modeling the variables, we will now introduce courses and classes for clauses and literals. For every clause K_i , we introduce a so-called clause-course \hat{c}_{K_i} . These courses contain three so-called literal-classes $\{c_{K_i}^1, c_{K_i}^2, c_{K_i}^3\}$. The upper indices of literal-classes indicate the index of the literal within its clause K_i . Formally,

$$\hat{\mathcal{C}}_K := \{\hat{c}_{K_i} = \{c_{K_i}^1, c_{K_i}^2, c_{K_i}^3\} \mid K_i \in \text{clause}(\varphi)\}.$$

We link a literal-class $c_{K_i}^j$ to $\hat{\delta}_{x_q}$ if literal j in clause K_i is \bar{x}_q . If literal j in clause K_i is x_q , we link the literal-class $c_{K_i}^j$ to $\hat{\delta}_{\bar{x}_q}$. Note the negation of the literal. This ensures that the right classes conflict within an organization. Summarizing, we set

$$\hat{\delta}(c_{K_i}^j) := \hat{\delta}_{\bar{l}_{i,j}} \quad \forall K_i \in \text{clause}(\varphi) \quad \forall j \in [3].$$

For each of the classes, we introduce a group consisting of a single unique base group. Each literal-class can be placed in weeks one, two or three using these groups. We link the groups to their corresponding clause-course:

$$\begin{aligned}\mathcal{G}_K &:= \{g_{K_i}^1, g_{K_i}^2, g_{K_i}^3 \mid K_i \in \text{clause}(\varphi)\} \\ \mathcal{G}_{\hat{c}}(\hat{c}_{K_i}) &:= \{g_{K_i}^1, g_{K_i}^2, g_{K_i}^3\} & \forall K_i \in \text{clause}(\varphi).\end{aligned}$$

o_{x_i}	w_1	w_2	w_3	w_4
x_i :	$c_{x_i} \quad g_{x_i}^1 \quad m$			$c_{x_i} \quad g_{x_i}^4 \quad m$

$o_{\bar{x}_i}$	w_1	w_2	w_3	w_4
x_i :	$c_{\bar{x}_i} \quad g_{x_i}^1 \quad m$			$c_{\bar{x}_i} \quad g_{x_i}^4 \quad m$

Figure 3.2: The two organizations resulting from a variable are shown as two tables. Each table has four weeks. Weeks two and three will become important later, when courses and classes for clauses and literals are introduced. Either both blue (dashed) or both red (dotted) assignments are valid in combination. The blue assignment represents an assignment of true to variable x_i as group $g_{x_i}^1$ is assigned to class c_{x_i} , resulting in a placement in week one of organization o_{x_i} . The red group assignment would result in an assignment of false to variable x_i .

Again, the week will be noted by the upper index of the group. Note that at least one literal-class of the clause has to be assigned group $g_{K_i}^1$, resulting in a placement in week one and fulfilling the clause. The eligible periods for clause-courses and their groups will have length 1. This is achieved by choosing an appropriate time interval.

$$\mathcal{P}_{\hat{c},g}(\hat{c}_{K_i}, g_{K_i}^j) := \{p_k^j = (w_j, d_1, [1])\} \quad \forall j \in [3] \quad \forall K_i \in \text{clause}(\varphi)$$

Rooms & Capacities The choice of eligible rooms does not matter. We assume a unique room exists for every class that could be assigned in the next step.

Using the capacity constraints we limit the classes that can be scheduled per week and organization. Since we only have a single day of week, both types of capacity constraints result in the same limitations if we set them equally. For all weeks and organizations, we set the capacities to m .

$$\text{cap}_{\text{weekly}}(o, w) = \text{cap}_{\text{daily}}(o, w, d_1) := m \quad \forall o \in \mathcal{O} \quad \forall w \in \mathcal{W}$$

To explain the output of the function $\text{avg}_{\mathcal{P},\mathcal{T}}$, central to the assessment of a combination of group and class, we need to take a look at the sets of eligible periods again. For all eligible combinations of course and group, the sets of eligible periods are confined to a single period featuring a single week and day. The function $\text{avg}_{\mathcal{P},\mathcal{T}}$ is always applied to the set of eligible periods of a course and one of its groups and the set of time slots for a week or date. Since all sets of eligible periods contain a single period confined to

3 Proposed Heuristic

a single day, $\text{avg}_{\mathcal{P},\mathcal{T}}$ will output the length of the period if the weeks of both sets match and 0. So $\text{avg}_{\mathcal{P},\mathcal{T}}$ will output m for variable-courses and 1 for clause-courses.

From the defined resources and functions, we construct the matching-space \mathcal{A} as described in [Definition 2.1](#).

Example We provide an example for the reduction from 3-SAT to APERIODIC CURRICULUM-BASED TIMETABLING - STEP 1 for the formula

$$\varphi = (x_1 \vee x_2 \vee x_1) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_1}).$$

Duplicate literals are eliminated to make the example easier to understand. Since the clauses now have a maximum of two literals, week three can be omitted from our construction as well. We show the reduction for the equivalent formula:

$$\varphi = (K_1 = (\ell_{1,1} = x_1 \vee \ell_{1,2} = x_2)) \wedge (K_2 = (\ell_{2,1} = \overline{x_1} \vee \ell_{2,2} = x_2)).$$

Firstly, we calculate which literals connect to which (combinations of) organizations:

$$\begin{array}{lll} K_1: & \hat{\delta}(c_{K_1}^1) = \hat{\delta}_{x_1} & \hat{\delta}(c_{K_1}^2) = \hat{\delta}_{x_2} \\ K_2: & \hat{\delta}(c_{K_2}^1) = \hat{\delta}_{x_1} & \hat{\delta}(c_{K_2}^2) = \hat{\delta}_{x_2} \end{array}$$

In [Figure 3.3](#), schematic timetables for the organizations resulting from the variables in φ are shown. Per column, we can either choose to schedule a long class connected to a variable or $m = 2$ classes connected to literals. Recall that the interesting week is week one. All variable-classes scheduled in week one result in an assignment of true to their respective variable.

A possible valid group assignment is shown by the classes marked by solid lines. Thus, clause K_1 is satisfied by its first literal, whereas clause K_2 is satisfied by its second literal. Variable x_1 is set to true because of assigning group $g_{x_1}^1$ to class c_{x_1} . Analogously, the same goes for variable x_2 , which is also set to true. This assignment obviously satisfies the formula.

Correctness To show the correctness of the given reduction, we will show that φ is satisfiable if and only if a feasible group assignment g exists. Note that daily capacities are equivalent to weekly capacities here because only a single day of week is used.

Recall that the classes that are scheduled in week one correlate with the assignment. For variable-classes, we will use the following correspondence between group assignment and variable assignment:

$$g(c_{x_i}) = g_{x_i}^1 \Leftrightarrow x_i \text{ is set to true.} \quad (3.1)$$

O_{x_1}	w_1	w_2	w_4
x_1 :	$\underline{c_{x_1} \quad g_{x_1}^1 \quad m=2}$		$\overline{c_{x_1} \quad g_{x_1}^4 \quad m=2}$
K_2 :	$\overline{c_{K_2}^1 \quad g_{K_2}^1 \quad 1}$	$\underline{c_{K_2}^1 \quad g_{K_2}^2 \quad 1}$	

$O_{\overline{x_1}}$	w_1	w_2	w_4
x_1 :	$\overline{c_{x_1} \quad g_{x_1}^1 \quad m=2}$		$\underline{c_{x_1} \quad g_{x_1}^4 \quad m=2}$
K_1 :	$\underline{c_{K_1}^1 \quad g_{K_1}^1 \quad 1}$	$\overline{c_{K_1}^1 \quad g_{K_1}^2 \quad 1}$	

O_{x_2}	w_1	w_2	w_4
x_2 :	$\underline{c_{x_2} \quad g_{x_2}^1 \quad m=2}$		$\overline{c_{x_2} \quad g_{x_2}^4 \quad m=2}$

$O_{\overline{x_2}}$	w_1	w_2	w_4
x_2 :	$\overline{c_{x_2} \quad g_{x_2}^1 \quad m=2}$		$\underline{c_{x_2} \quad g_{x_2}^4 \quad m=2}$
K_1 :	$\overline{c_{K_1}^2 \quad g_{K_1}^1 \quad 1}$	$\underline{c_{K_1}^2 \quad g_{K_1}^2 \quad 1}$	
K_2 :	$\underline{c_{K_2}^2 \quad g_{K_2}^1 \quad 1}$	$\overline{c_{K_2}^2 \quad g_{K_2}^2 \quad 1}$	

Figure 3.3: A graphical representation of the schedules of the example. Each column represents a week and each row the group choices for a class. On top of each bar, representing a class, first the name of the class is written, then the group and lastly the length of the class, which is equal to the output of the function $\text{avg}_{\mathcal{P}, \mathcal{T}}$.

3 Proposed Heuristic

(\Rightarrow) Assume we have a valid assignment that satisfies our formula φ . We show that the constructed group assignment g is a feasible group assignment, meaning every class is assigned a group, per course all groups are assigned to a class, and no capacities are violated.

We assign groups to all variable-classes using [Equation \(3.1\)](#). Week one of all organizations belonging to the variable-class that is assigned the group with upper index one is at the limit of its capacity then. Thus, no literal-classes can be assigned to this organization in week one. Each literal-class needs to be assigned a group, however. This is because we have three literal-classes per clause-course and three eligible groups. So one literal-class needs to be assigned a group with upper index one. Therefore, this literal-class takes care of satisfying its clause. Otherwise, no feasible group assignment can be found.

(\Leftarrow) Assume we have a feasible group assignment g constructed by the described construction. We show that the corresponding assignment satisfies the formula: An assignment satisfies a formula in 3-CNF if and only if in every clause at least one literal evaluates to true.

We use [Equation \(3.1\)](#) to construct an assignment for our formula φ . Since in every clause-course at least one literal-class is assigned to week one, all clauses are satisfied. \square

3.1.2 Integer Linear Program

To this end we use an ILP formulation to solve APERIODIC CBT - STEP 1.

Variables For every eligible combination of class and group, we will use a binary variable. A value of 1 will indicate the chosen assignment:

$$X_{c,g} \in \{0, 1\} \quad \forall c \in \mathcal{C} \quad \forall g \in \mathcal{G}_{\hat{c}}(\hat{c}(c)).$$

Objective Though the decision problem is only concerned with finding a valid *group assignment*, the ILP stated here will respect the preferences made by the organizations for time slots.

$$\min \sum_{c \in \mathcal{C}} \sum_{g \in \mathcal{G}_{\hat{c}}(\hat{c}(c))} X_{c,g} \sum_{o \in \delta(c)} \sum_{p \in \mathcal{P}_{\hat{c},g}(\hat{c}(c),g(c))} \sum_{t \in p} \text{pref}_{o,t}(o, t)$$

Note that the values following the decision variable can be precomputed.

Essential Constraints First, we have to make sure, that every class is assigned to exactly one group. In the second constraint, we guarantee that every group of a course gets exactly one class. These two constraints together ensure a bijection between classes

and groups per course.

$$\sum_{g \in \mathcal{G}_{\hat{c}}(\hat{c}(c))} X_{c,g} = 1 \quad \forall c \in \mathcal{C} \quad (\text{taking place (1)})$$

$$\sum_{c \in \hat{c}} X_{c,g} = 1 \quad \forall \hat{c} \in \hat{\mathcal{C}} \quad \forall g \in \mathcal{G}_{\hat{c}}(\hat{c}) \quad (\text{group obligation (2)})$$

Capacity Constraints To approximately stay below the capacities of organizations, we sum up the average load of all class group combinations belonging to an organization and time frame. Then, we keep this sum below capacity:

$$\sum_{c \in \mathcal{C}_o(o)} \sum_{g \in \mathcal{G}_{\hat{c}}(\hat{c}(c))} \text{avg}_{\mathcal{P}, \mathcal{T}}(\mathcal{P}_{\hat{c}, g}(\hat{c}(c), g), \mathcal{T}(w)) \cdot X_{c,g} \leq \text{cap}_{\text{weekly}}(o, w) \quad \forall w \in \mathcal{W} \quad \forall o \in \mathcal{O} \quad (\text{weekly capacity (3)})$$

$$\sum_{c \in \mathcal{C}_o(o)} \sum_{g \in \mathcal{G}_{\hat{c}}(\hat{c}(c))} \text{avg}_{\mathcal{P}, \mathcal{T}}(\mathcal{P}_{\hat{c}, g}(\hat{c}(c), g), \mathcal{T}(w) \cap \mathcal{T}(d)) \cdot X_{c,g} \leq \text{cap}_{\text{daily}}(o, w, d) \quad \forall w \in \mathcal{W} \quad \forall d \in \mathcal{D} \quad \forall o \in \mathcal{O} \quad (\text{daily capacity (4)})$$

The result is a matching between classes and groups. We will use the already defined function $g: \mathcal{C} \rightarrow \mathcal{G}$ to refer to the result of this step.

3.2 Step 2

In the last step, a period and room are assigned to every class (see [Figure 3.4](#)). We assume that we are given a feasible group assignment. The intention of this step is to produce a schedule free of intersections in time for every base group and every room. As before, we also need to take organizational capacities into account.

Now, we will define the constraints a feasible *period & room assignment* has to obey. Each class has to be assigned an eligible room and period.

We will refer to the result by the two functions $p: \mathcal{C} \rightarrow \mathcal{P}$ and $r: \mathcal{C} \rightarrow \mathcal{R}$.

Definition 3.3. We call $\mathcal{L} \subseteq \mathcal{A}$ a feasible *period & room assignment* if

$$\forall c \in \mathcal{C} \quad \forall t \in p(c) \quad \forall b \in g(c): (c, t, r(c), b) \in \mathcal{L} \quad (\text{taking place (1)})$$

$$\forall e \in E(\mathbb{H}_{\text{conf}}): \mathcal{L} \cap e \neq e \quad (\text{conflicts (3)})$$

Both constraints follow directly from APERIODIC CBT. The [group obligation \(2\)](#) constraint is missing since we took care of that in the previous step. Recall that room conflicts, group conflicts and organizational capacities are all encoded in the hypergraph \mathbb{H}_{conf} and thus part of the definition.

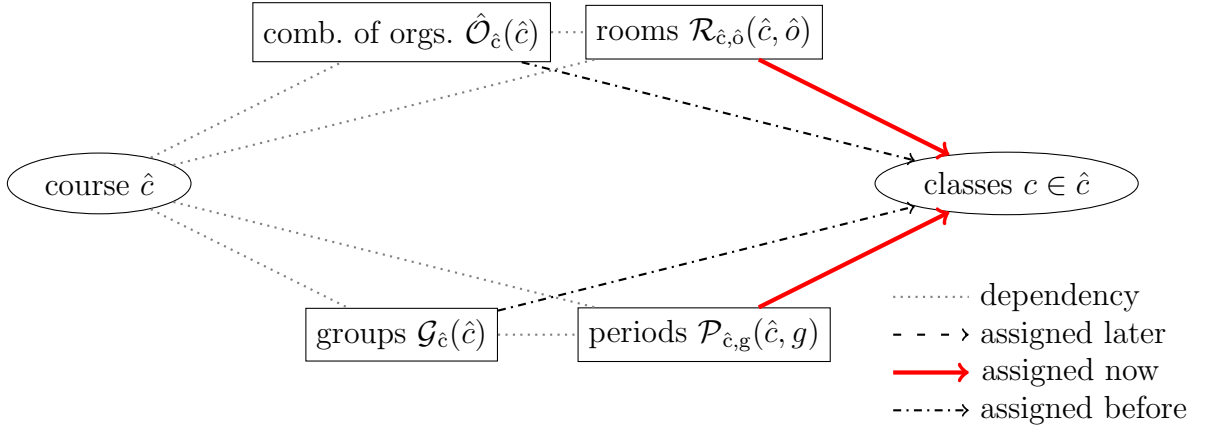


Figure 3.4: Assigning a period and room to each class. Highlighted as solid red arrows are the assignments of a period and a room to a class, performed in this step. The combination of organizations and the group have already been assigned in Section 2.2 and Section 3.1, respectively. For a more detailed description refer to Figure 2.3.

APERIODIC CURRICULUM-BASED TIMETABLING - STEP 2

Input: The set \mathcal{A} of all possible assignments, the function $\hat{o}: \mathcal{C} \rightarrow \hat{\mathcal{O}}$ from Section 2.2 and a feasible group assignment $g: \mathcal{C} \rightarrow \mathcal{G}$ from Section 3.1.

Task: Find a feasible *period & room assignment* p and r .

3.2.1 Integer Linear Program

Again, we use an ILP formulation to solve APERIODIC CBT - STEP 2. The overall idea follows along the *Intuitive Integer Program* shown by Lach and Lübbecke [LL08].

To simplify further definitions, we will define two functions that use the results from Section 2.2 and Section 3.1:

$$\begin{aligned}
 \mathcal{P}_c: \mathcal{C} &\rightarrow 2^{\mathcal{P}} & c &\mapsto \mathcal{P}_{\hat{c},g}(\hat{c}(c), g(c)) \cap \bigcap_{o_i \in \hat{o}(c)} \mathcal{P}_o(o_i) \\
 \mathcal{R}_c: \mathcal{C} &\rightarrow 2^{\mathcal{R}} & c &\mapsto \mathcal{R}_{\hat{c},\hat{o}}(\hat{c}(c), \hat{o}(c))
 \end{aligned}$$

Variables For every eligible class period combination, we will use a binary variable. A value of 1 will indicate the chosen period.

$$Y_{c,p} \in \{0, 1\} \quad \forall c \in \mathcal{C} \quad \forall p \in \mathcal{P}_c(c)$$

Another set of variables represents all eligible class period room combinations. Again a value of 1 will indicate the chosen period and room. The decisions of both sets of variables have to match per class. Ambiguity will be ruled out by the constraints.

$$Z_{c,p,r} \in \{0, 1\} \quad \forall c \in \mathcal{C} \quad \forall p \in \mathcal{P}_c(c) \quad \forall r \in \mathcal{R}(c)$$

Objective Whereas the decision problem is only concerned with finding a feasible schedule, here we will take into account the time preferences of organizations. Also per course, room and combination of organizations room preferences will be respected here.

$$\begin{aligned} \min & \sum_{c \in \mathcal{C}} \sum_{p \in \mathcal{P}_c(c)} Y_{c,p} \sum_{o \in \hat{o}(c)} \sum_{t \in p} \text{pref}_{o,t}(o, t) \\ & + \sum_{c \in \mathcal{C}} \sum_{p \in \mathcal{P}_c(c)} \sum_{r \in \mathcal{R}_c(c)} Z_{c,p,r} \cdot \text{pref}_{\hat{c},r,\hat{o}}(\hat{c}(c), r, \hat{o}(c)) \end{aligned}$$

Note that the values following the decision variables can be precomputed and thus the objective function is indeed linear.

Essential Constraints Firstly, we ensure that every class has exactly one assigned period and room by forcing the sum of Y variables per class to one. Also, we connect the period and room decision variables by forcing the sum over eligible room decisions to be equal to the period decision variable per class and period.

$$\sum_{p \in \mathcal{P}_c(c)} Y_{c,p} = 1 \quad \forall c \in \mathcal{C} \quad (\text{period assignment (1)})$$

$$\sum_{r \in \mathcal{R}(c)} Z_{c,p,r} = Y_{c,p} \quad \forall c \in \mathcal{C} \quad \forall p \in \mathcal{P} \quad (\text{room assignment (2)})$$

Next, we prohibit every room and group conflict taken from our conflict graph. In case of room conflicts, we limit the classes for every time slot room combination to one.

$$\sum_{p_1 \in \mathcal{P}_c(c_1) \cap \mathcal{P}(t) \cap \mathcal{P}_r(r)} Z_{c_1,p_1,r} + \sum_{p_2 \in \mathcal{P}_c(c_2) \cap \mathcal{P}(t) \cap \mathcal{P}_r(r)} Z_{c_2,p_2,r} \leq 1 \quad \forall (c_1, t, r, b_1), (c_2, t, r, b_2) \in E_{\mathcal{R}} \quad (\text{no room conflicts (3)})$$

Group conflicts are avoided by limiting the classes per base group and time slot to one.

$$\sum_{p_1 \in \mathcal{P}_c(c_1) \cap \mathcal{P}(t)} Y_{c_1,p_1} + \sum_{p_2 \in \mathcal{P}_c(c_2) \cap \mathcal{P}(t)} Y_{c_2,p_2} \leq 1 \quad \forall (c_1, t, r_1, b), (c_2, t, r_2, b) \in E_{\mathcal{G}} \quad (\text{no group conflicts (4)})$$

Organizational Capacities Constraints To respect the weekly and daily capacities of organizations, we count the minutes over all time slots of a week and date, respectively, limiting them by the stated capacity. For parallel capacities, we instead count the

3 Proposed Heuristic

number of classes per time slot.

$$\sum_{c \in \mathcal{C}_o(o)} \sum_{p \in \mathcal{P}_c(c) \cap \mathcal{P}(w)} Y_{c,p} \cdot \sum_{t \in p \cap \mathcal{T}(w)} \text{length}_t(t) \leq \text{cap}_{\text{weekly}}(o, w) \quad \forall o \in \mathcal{O} \quad \forall w \in \mathcal{W} \quad (\text{weekly capacity (5)})$$

$$\sum_{c \in \mathcal{C}_o(o)} \sum_{p \in \mathcal{P}_c(c) \cap \mathcal{P}(w) \cap \mathcal{P}(d)} Y_{c,p} \cdot \sum_{t \in p \cap \mathcal{T}(w) \cap \mathcal{T}(d)} \text{length}_t(t) \leq \text{cap}_{\text{daily}}(o, w, d) \quad \forall o \in \mathcal{O} \quad \forall w \in \mathcal{W} \quad \forall d \in \mathcal{D} \quad (\text{daily capacity (6)})$$

$$\sum_{c \in \mathcal{C}_o(o)} \sum_{p \in \mathcal{P}_c(c) \cap \mathcal{P}(t)} Y_{c,p} \leq \text{cap}_{\text{parallel}}(o, t) \quad \forall o \in \mathcal{O} \quad \forall t \in \mathcal{T} \quad (\text{parallel capacity (7)})$$

3.2.2 Room Set Contraction

Up until now, we have been creating a variable for every possible period room combination per class. This is standard procedure for most algorithms solving CBT, at least when using Integer Programming. But when we take a look back at the input, we notice that the set of eligible rooms is not unique per class. Instead, it is given per course and combination of organizations. We will use this to reduce the number of variables by not using a class as the first index of our Z variables anymore.

The obvious replacement for the first index would be course and combination of organizations but we can save even more variables. By using artificially crafted *room sets* we can collapse more classes into the same set of variables. The idea is to group all classes together that share the same room requirements and thus have the same set of eligible rooms.

So for all classes with the same set of eligible rooms, we create a room set \hat{r} . For this room set a variable for each eligible period is created. Then, for every time slot we guaranteed that enough rooms are available for all classes that are assigned to a period containing said time slot.

$$\hat{\mathcal{R}} := \bigcup_{\hat{c} \in \hat{\mathcal{C}}} \bigcup_{\hat{d} \in \hat{\mathcal{O}}_{\hat{c}}(\hat{c})} \{\hat{r} \subseteq \mathcal{R} \mid \hat{r} = \mathcal{R}_{\hat{c}, \hat{d}}(\hat{c}, \hat{d})\}$$

We denote by $\mathcal{C}_{\hat{r}}: \hat{\mathcal{R}} \rightarrow 2^{\mathcal{C}}$ the classes belonging to a room set and by $\hat{\mathcal{R}}_{\hat{r}}: \mathcal{R} \rightarrow 2^{\hat{\mathcal{R}}}$ the room sets a room is part of. Also $\hat{\mathcal{R}}_{\hat{c}}: \hat{\mathcal{C}} \rightarrow 2^{\hat{\mathcal{R}}}$ gives a set of room sets a course has classes in and $\hat{r}_c: \mathcal{C} \rightarrow \hat{\mathcal{R}}$ gives the single room set a class belongs to.

Variables We substitute the variables for each eligible class, period and room with variables per room set, period and room. Our modified variables Z' stay binary because we cannot schedule more than one class per period room combination anyway.

$$Z'_{\hat{r}, p, r} \in \{0, 1\} \quad \forall \hat{r} \in \hat{\mathcal{R}} \quad \forall p \in \mathcal{P} \quad \forall r \in \hat{r}$$

Objective We need to modify our objective function to accommodate for this change. While the first line stays unmodified, the second needs to aggregate the preferences of individual classes contained in the room set.

$$\begin{aligned} \min & \sum_{c \in \mathcal{C}} \sum_{p \in \mathcal{P}_c(c)} Y_{c,p} \sum_{o \in \hat{o}(c)} \sum_{t \in \mathcal{P}} \text{pref}_{o,t}(o, t) \\ & + \sum_{\hat{c} \in \hat{\mathcal{C}}} \sum_{\hat{r} \in \hat{\mathcal{R}}_{\hat{c}}(\hat{c})} \sum_{r \in \hat{r}} \sum_{p \in \mathcal{P}_r(r)} Z'_{\hat{r},p,r} \cdot \sum_{c_i \in \mathcal{C}_{\hat{r}}(\hat{r})} \text{pref}(\hat{c}(c_i), r, \hat{o}(c_i)) \end{aligned}$$

Essential Constraints The constraints are also affected by this change. In constraint **room assignment (2')**, we guarantee that enough rooms of the room set are available for all the classes scheduled in a period. Constraint **no room conflicts (3')** sees no major changes other than accounting for the index change.

$$\begin{aligned} \sum_{r \in \hat{r}} Z'_{\hat{r},p,r} &= \sum_{c \in \mathcal{C}_{\hat{r}}(\hat{r})} Y_{c,p} \\ &\forall \hat{r} \in \hat{\mathcal{R}} \quad \forall p \in \mathcal{P} \\ &\text{(room assignment (2'))} \\ \sum_{p_1 \in \mathcal{P}_c(c_1) \cap \mathcal{P}(t) \cap \mathcal{P}_r(r)} Z'_{\hat{r}_c(c_1),p_1,r} &+ \sum_{p_2 \in \mathcal{P}_c(c_2) \cap \mathcal{P}(t) \cap \mathcal{P}_r(r)} Z'_{\hat{r}_c(c_2),p_2,r} \leq 1 \\ &\forall (c_1, t, r, b_1), (c_2, t, r, b_2) \in E_{\mathcal{R}} \\ &\text{(no room conflicts (3'))} \end{aligned}$$

3.3 Performance & Benchmarks

A real-world implementation of the described heuristic exists within the planning tool *Moses*. The model and heuristic described in this thesis only cover the basic framework of the algorithm, though.

3.3.1 Real World Implementation Differences

Many constraints that surpass the scope of this thesis exist to assist planning large programs. The differences of the real-world implementation in *Moses* will be described in the following paragraphs.

Soft Constraints Some constraints that are modeled as hard constraints in this thesis are implemented as soft constraints. This is done by the introduction of a so-called penalty variable per constraint. These penalty variables are then added to the objective function with a large enough coefficient for the solver to avoid assigning any value other than 0, thus not violating the original constraint. The weighting of the coefficients of the penalty variables can then be tuned to bias the solution in one direction or another.

The use of soft constraints comes from the fact that for the planner a schedule violating a few constraints is still more valuable than no schedule at all when the problem

3 Proposed Heuristic

is infeasible. Therefore, assigning a period, a room and organizational capacities are handled as soft constraints in both steps of the heuristic. Depending on the importance of the organizations' capacities not being exceeded it can be pondered whether it is more desirable to violate the capacities or not to assign the class at all.

Other Constraints As already mentioned, many other constraints exist within the real-world implementation, all of which are modeled as soft constraints. Some examples are:

- Transit times between classes
- Chronological orderings of courses or classes
- Minimum and maximum of classes to be held in parallel per course
- Minimum and maximum time gaps between courses or classes
- Minimum and maximum number of classes per date of a set of courses
- Minimum and maximum number of dates a set of classes should span
- Holding different classes in the same room
- Compactness for students/groups

Step 1.5 The transit time and chronological ordering constraints can be used to reduce the search space, resulting in much better performance. In the following, we will discuss a further step in the heuristic applied between steps 1 and 2 described in this thesis.

Chronological orderings state in which order courses should be attended by students. One way to achieve this is to already limit the available periods of each course during the input phase in a way that guarantees the desired succession. This, however, would be a lot of work for the person entering the data and therefore Moses offers the ability to state orderings on courses without explicitly limiting their eligible periods. These constraints are then modeled within the conflict graph by prohibiting every combination that violates the given order.

Another constraint used to limit the search space are transit times between classes. Every student must have enough time to travel from one class to the next because sometimes switching classes involves changing campuses. To avoid long transit times campus changes are to be avoided or kept low.

These two circumstances together are used in the already advertised, additional step between 1 and 2. We will refer to this step as **APERIODIC CBT - STEP 1.5** in the following. There the goal is to assign each base group to a campus on a given date. If we restrict the eligible rooms for every class to not span more than one campus, we can then assign classes to dates while respecting the ordering and transit constraints. Thus, the set of eligible periods can be severely reduced to be within the assigned date. This simultaneously reduces the set of eligible periods per class and avoids campus changes.

Table 3.1: Some statistics for the samples used, where $|\hat{\mathcal{C}}|$ is the number of courses and $|\mathcal{C}|$ the number of classes. Per course, the average number of groups is denoted by $\text{avg}|\mathcal{G}_{\hat{c}}(\hat{c})|$ which is relevant for Step 1. For Step 2, the average number of eligible periods and rooms per class are denoted by $\text{avg}|\mathcal{P}_c(c)|$ and $\text{avg}|\mathcal{R}_c(c)|$, respectively. Finally, assigned $|\mathcal{C}|$ in % states the percentage of classes that could be assigned a group, period and room after both steps.

Dataset	$ \hat{\mathcal{C}} $	$ \mathcal{C} $	$\text{avg} \mathcal{G}_{\hat{c}}(\hat{c}) $	$\text{avg} \mathcal{P}_c(c) $	$\text{avg} \mathcal{R}_c(c) $	assigned $ \mathcal{C} $ in %
<i>trad+ref0</i>	158	259	1.64	61.87	9.75	98.07
<i>trad+ref2</i>	501	3,271	6.53	64.59	13.67	98.69
<i>trad+ref4</i>	799	8,095	10.13	65.35	13.42	98.68
<i>trad+ref6</i>	1,030	11,069	10.75	63.93	13.09	98.60
<i>trad+ref8</i>	1,305	14,104	10.81	69.55	12.09	98.77
<i>trad+ref10</i>	1,447	15,353	10.61	69.92	11.49	98.80

3.3.2 Benchmarks

To provide benchmarks we use the data of a medical university in Germany with five programs that are to be planned using Moses. Four of these programs are structured very traditionally with only periodic courses and have three to five semesters each. Every semester has between 15 and 70 students and between 4 and 6 periodic courses that take place in every week of the semester. The last one, however, is a newly designed reform program featuring aperiodic courses almost exclusively. It consists of 10 semesters with more than 300 students each. The teaching format with the least student per class requires three students. Therefore, base groups have three students, resulting in about 100 base groups per semester. These base groups form different groups with different sizes, ranging from three students, for classes with patients, to more than 300 for lectures.

We used different samples of real data from the mentioned university to test the proposed heuristic. We will refer to each sample by a name descriptive of its content: *trad+refX* contains all semesters of the traditional programs plus the first X semesters of the reform program, where $X \in \{0, 2, 4, 6, 8, 10\}$.

All constraints not included in this paper were disabled where possible. Still, the organizational capacities and period and room assignment are soft constraints. Otherwise, we would only have had infeasible problems using the real world data from the mentioned university. The percentage of classes that could be assigned a period and room are therefore included in Table 3.1, where we present some statistics on the samples we used for benchmarking. The measures are after applying steps 1, 1.5 and 2. Unfortunately it was not possible to get any results within reasonable time from Step 2 without applying the briefly described Step 1.5 first.

Hardware For all runs, the same computer running the same configuration was used. The CPU is an *Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz* with around 140GB of memory available. Our runs were restricted to using 8 threads and a maximum of 64GB of memory. The Gurobi solver software in version 8.0.1 was used on top of Linux Kernel

3 Proposed Heuristic

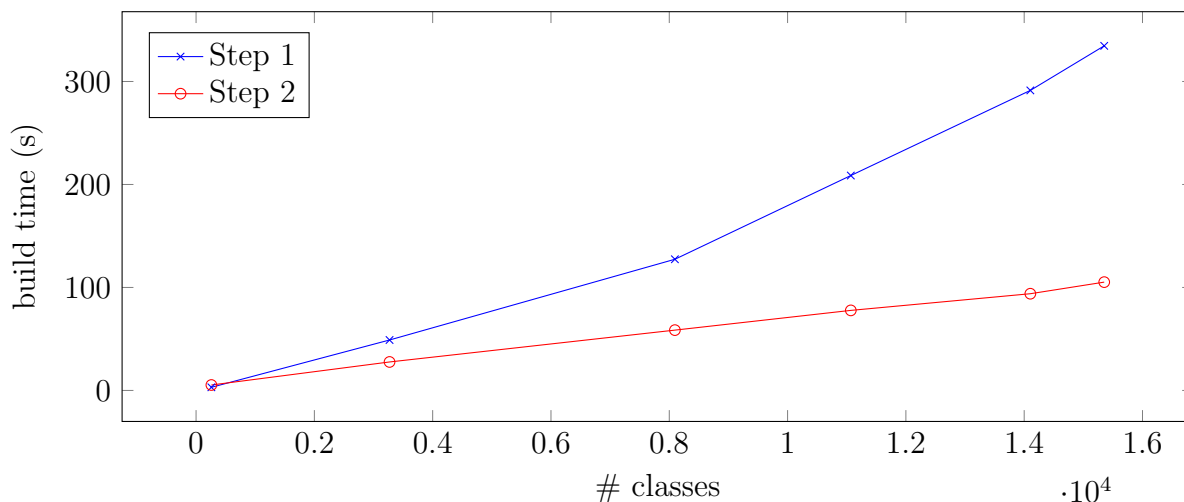


Figure 3.5: Time spent building the ILP model as a function of the number of classes.

version 2.6.32.

Interpretation Table 3.1 clearly shows that the traditional programs have far fewer students and therefore fewer base groups. This also results in fewer groups per teaching format and consequently fewer classes per course. The average amount of eligible periods and rooms however stay roughly constant across the programs.

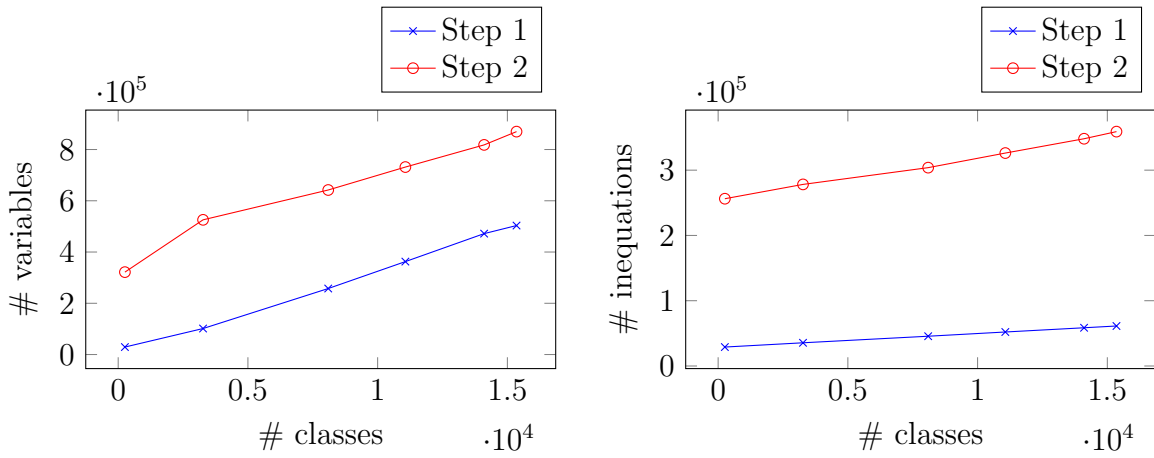
Interestingly, building the model for APERIODIC CBT - STEP 1 is more time consuming than for APERIODIC CBT - STEP 2 as seen in Figure 3.5.

This stands in contrast to Step 2 having significantly more variables and inequations as can be seen in Figures 3.6a and 3.6b. This discrepancy is mainly due to the fact that in Step 1, the values of the function $\text{avg}_{\mathcal{P},\mathcal{T}}$ have to be precomputed for all combinations of class and week in the case of weekly capacities and for all combinations of class, week and day in the case of daily capacities. Still, building times seem almost linear in the number of classes.

Likewise for solving, increasing the number of classes impairs the time spent almost linearly for Step 1 as seen in Figure 3.7. Surprisingly, for Step 2 solving the *trad+ref6* instance with 11,069 classes takes more time than the larger *trad+ref8* instance with 14,104 classes. This may be due to some unaccounted differences in input or the Gurobi solver choosing a different strategy.

For Step 1, the average number of groups and therefore classes per course correlates loosely with the time spent solving the step as seen in Figure 3.8. Overall size as in number of courses and classes does seem to have a larger effect, though. In *trad+ref4* and above, the average number of groups per course stays roughly constant but the time spent solving increases further.

In Step 2 neither the average number of periods nor rooms seem to have any impact on the time spent solving as seen in Figure 3.9.



(a) Number of variables as a function of the (b) Number of inequations as a function of the number of classes.

Figure 3.6: Size of the ILP models.

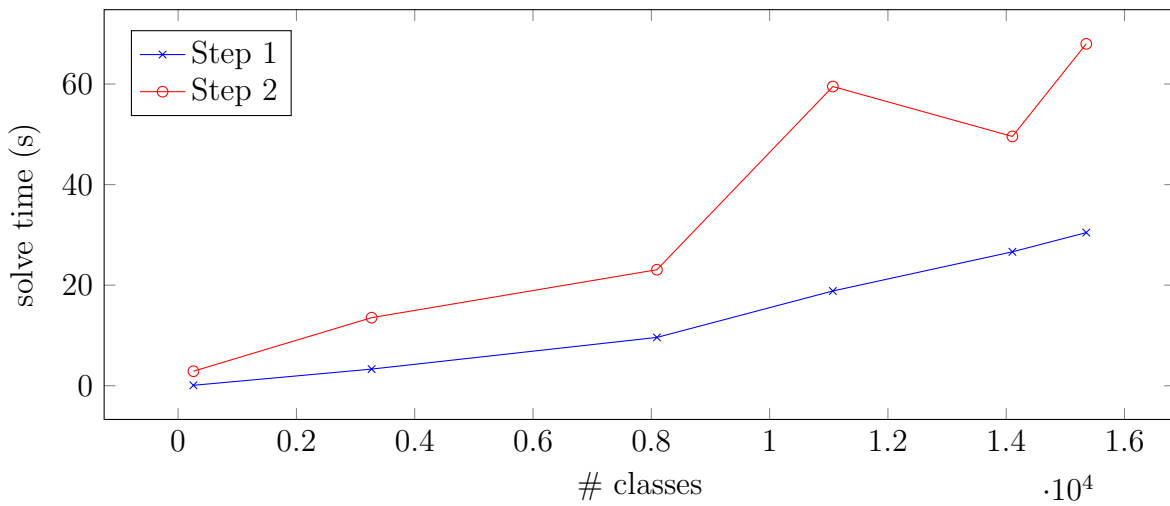


Figure 3.7: Time spent in seconds solving APERIODIC CBT as a function of the number of classes.

3 Proposed Heuristic

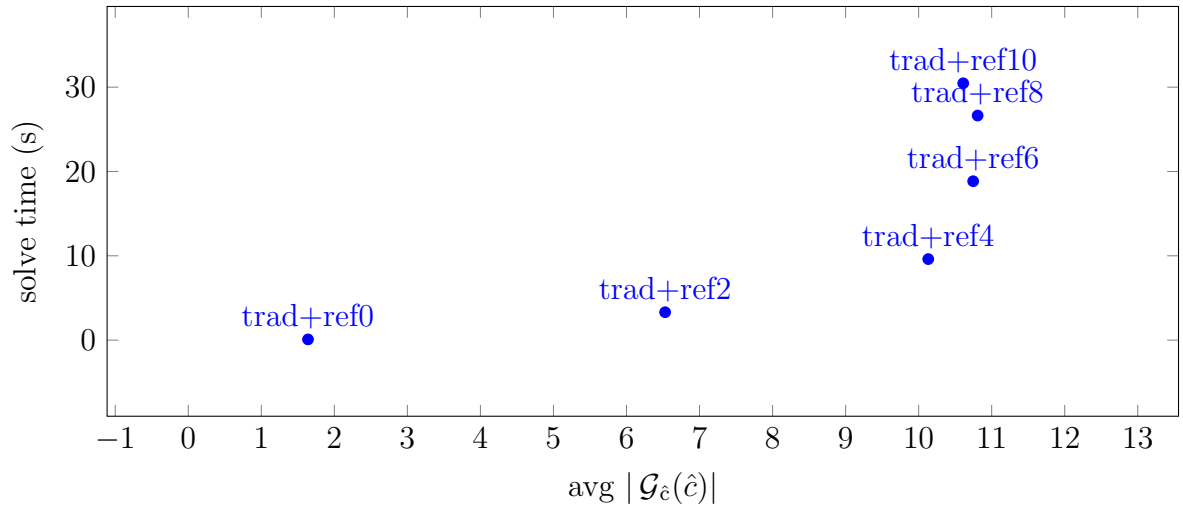


Figure 3.8: Time spent in seconds solving APERIODIC CBT - STEP 1 as a function of the average number of groups per course.

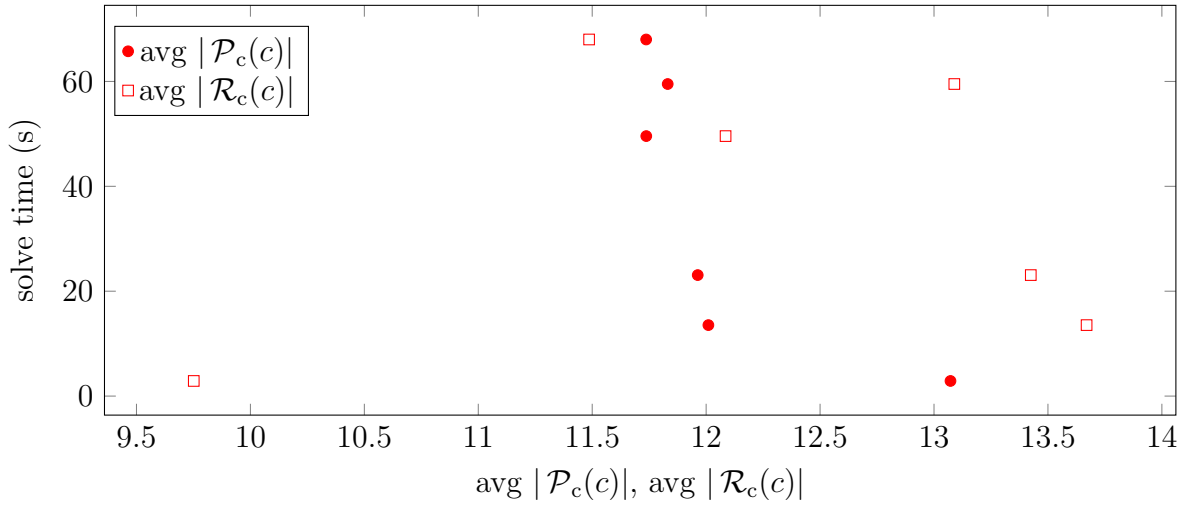


Figure 3.9: Time spent in seconds solving APERIODIC CBT - STEP 2 as a function of the average number of eligible periods per class and the average number of eligible rooms per class.

4 Conclusion & Outlook

We described challenges of automatically generating timetables for universities employing reform programs. To that end, the necessary additions to *Curriculum-Based Timetabling (CBT)* have been worked out and a new model called *Aperiodic Curriculum-Based Timetabling (Aperiodic CBT)* has been developed. We showed that solving Aperiodic CBT naïvely would probably not work because of the size of the involved conflict graph. Therefore, we proposed a heuristic that solves the problem in two steps.

As can be seen in [Section 3.3.2](#), the proposed heuristic works quite well for the real-world data used. The plots show that running time is not a problem, yet. Thus, it seems likely that the heuristic can still perform well with bigger problem instances, despite being NP-complete. However, it remains unclear if the heuristic provides close-to-optimum solutions as no approximation guarantees are given. We conclude this thesis with some further topics of future interest.

NP-Hardness of Step 1 Despite proving in [Section 3.1.1](#) that the decision problem version of Step 1 of the heuristic is in fact NP-complete, all test runs done for the benchmarks immediately produced an integer solution for the relaxation of the integer program. This may be due to some unaccounted structure in the input data or some clever implementation on the side of the Gurobi solver. Reformulating the problem to exploit more structure might as well lead to a computationally less demanding algorithm for Step 1.

Integrating Further Constraints Integrating all in [Section 3.3.1](#) briefly described constraints into the problem definition might prove interesting, too. Some of the constraints can surely be integrated into the conflict graph already used to describe room, group and capacity constraints. Others may define constraint classes on their own. Since all these constraints refer either to times or rooms, they will have to be implemented in Step 2 of the heuristic. It remains to be seen how they would be implemented in the ILP, even if they can be integrated into the conflict graph.

Investigating Step 1.5 Possible implementations and computational complexity of Step 1.5 of the heuristic (see [Section 3.3.1](#)) are also up for discussion. Recall that, in Step 1.5, classes are assigned to sets of dates to narrow down the matching-space for Step 2. Without at least the further constraints of [Section 3.3.1](#), the assignment of classes to dates becomes a guessing game and may destroy some good solutions otherwise to be found in Step 2.

Decomposition of Step 2 One idea to get rid of Step 1.5 might be a decomposition of Step 2 as done by Lach and Lübbecke [LL08] for CBT. Since Step 2 of the heuristic is quite similar to CBT, this might be possible without much need for change. Lach and Lübbecke [LL08] first assign periods to all classes (lectures) taking care of rooms by scheduling no more classes than the number of fitting rooms per period. Then, rooms are assigned in a second step, which can be performed in polynomial time. Their approach provides exact solutions. This could reduce the complexity of Step 2 to a manageable level without performing Step 1.5 first, as was necessary in our benchmarks.

Literature

- [Abu10] F. N. Abu-Khazam. “A kernelization algorithm for d -Hitting Set”. In: *J. Comput. Syst. Sci.* 76.7 (2010), pp. 524–531. URL: <https://doi.org/10.1016/j.jcss.2009.09.002> (cit. on p. 24).
- [Bon+12] A. Bonutti, F. D. Cesco, L. D. Gaspero, and A. Schaerf. “Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results”. In: *Annals OR* 194.1 (2012), pp. 59–70. URL: <https://doi.org/10.1007/s10479-010-0707-0> (cit. on p. 29).
- [HL16] J. Höner and G. Lach. “Post-enrollment-based course timetabling with Moses: System demonstration”. In: *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling*. 2016 (cit. on p. 10).
- [KS13] S. Kristiansen and T. R. Stidsen. *A Comprehensive Study of Educational Timetabling - a Survey*. Report 978-87-93130-02-9. Department of Management Engineering, Technical University of Denmark, 2013 (cit. on p. 7).
- [LL08] G. Lach and M. E. Lübbecke. “Optimal University Course Timetables and the Partial Transversal Polytope”. In: *Experimental Algorithms, 7th International Workshop, WEA 2008, Proceedings*. Vol. 5038. Lecture Notes in Computer Science. Springer, 2008, pp. 235–248. URL: https://doi.org/10.1007/978-3-540-68552-4_18 (cit. on pp. 38, 48).
- [LL12] G. Lach and M. E. Lübbecke. “Curriculum based course timetabling: new solutions to Udine benchmark instances”. In: *Annals OR* 194.1 (2012), pp. 255–272. URL: <https://doi.org/10.1007/s10479-010-0700-7> (cit. on p. 29).
- [LLZ16a] G. Lach, M. Lach, and E. Zorn. “Examination timetabling with Moses: System demonstration”. In: *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling*. 2016 (cit. on p. 11).
- [LLZ16b] G. Lach, M. Lach, and E. Zorn. “University course timetabling with Moses: System demonstration”. In: *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling*. 2016 (cit. on p. 10).

Literature

- [McC+10] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke. “Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition”. In: *INFORMS Journal on Computing* 22.1 (2010), pp. 120–130. URL: <https://doi.org/10.1287/ijoc.1090.0320> (cit. on p. 29).
- [Sch99] A. Schaerf. “A Survey of Automated Timetabling”. In: *Artif. Intell. Rev.* 13.2 (1999), pp. 87–127. URL: <https://doi.org/10.1023/A:1006576209967> (cit. on pp. 7, 29).
- [SLH08] C. Stosch, K. A. Lehmann, and S. Herzig. “Time for Change - Die Implementierung des Modellstudiengangs Humanmedizin in Köln”. In: *Zeitschrift für Hochschulentwicklung* 3.3 (Dec. 9, 2008). URL: <http://www.zfhe.at/index.php/zfhe/article/view/68> (cit. on p. 8).
- [Wer85] D. de Werra. “An introduction to timetabling”. In: *European Journal of Operational Research* 19.2 (Feb. 1, 1985), pp. 151–162. URL: <http://www.sciencedirect.com/science/article/pii/0377221785901675> (cit. on p. 7).
- [Cha] Charité. *Charité Universitätsmedizin Berlin - Modellstudiengang Humanmedizin*. URL: https://www.charite.de/studium_lehre/studiengaenge/modellstudiengang_humanmedizin/ (visited on 02/25/2019) (cit. on p. 8).
- [Mata] MathPlan. *Automatisierte Stundenplanung | MathPlan*. URL: <https://www.mathplan.de/projekte.html> (visited on 02/25/2019) (cit. on p. 9).
- [Matb] MathPlan. *Über Uns | MathPlan*. URL: <https://www.mathplan.de/ueber-uns.html> (visited on 05/21/2019) (cit. on p. 9).
- [RWT] RWTH. *RWTH Aachen - Überschneidungsfreie Stundenpläne*. URL: <https://www.or.rwth-aachen.de/de/praxis-details/carpe-diem.html> (visited on 02/25/2019) (cit. on pp. 9, 10).
- [TUB] TUB. *TUB innoCampus: Moses*. URL: <https://www.innocampus.tu-berlin.de/projekte/moses/> (visited on 05/21/2019) (cit. on p. 9).
- [TUM] TUM. *TUM IT - CIO: Moses*. URL: <https://www.it.tum.de/projekte/moses/> (visited on 02/25/2019) (cit. on p. 10).