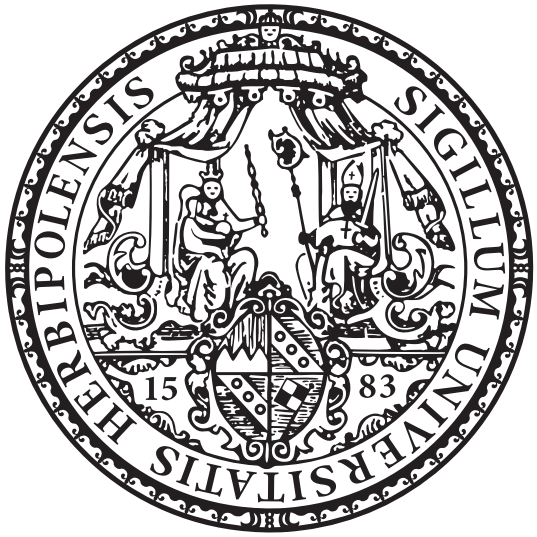
Bachelor Thesis

Reducing cancelled lessons by improving timetables with stochastic optimization

Andreas Maier

Date of Submission: 28. August 2023
Advisor: Prof. Dr. Marie Schmidt



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Abstract

The construction of timetables is a problem that is relevant for many applications, most notably for educational institutions. While the timetable of schools reflects how most of the day and the week is going to unfold in terms of which courses take place when, where and by whom, spontaneous and unforeseeable absences are common, disturbing the schedule defined by the timetable. For that reason, at schools and other institutions that cannot or do not want to simply drop the affected lessons on the cost of the students, substitution plans are supposed to regulate these occurrences. However, if a timetable can be well designed in general, but when it comes to substitution plans, which actually reflect how the day is going to unfold, these might be suboptimal with respect to how good the quality of replacement lessons is. This suboptimality could potentially be eliminated if substitution plans are considered at the time of the timetables construction. The aim of this work is to explore this aspect of optimizing timetables more closely, presenting a stochastic integer linear program to experimentally examine how promising such an optimization can be.

Zusammenfassung

Das Erstellen von Stundenplänen ist ein Problem das für viele Anwendungen relevant ist, allen voran für Bildungsinstutionen. Während Stundenpläne von Schulen eine Reflexion davon sind, wie der größte Teil des Tages sich voraussichtlich abspielen wird im Sinne von wann welche Kurse wo stattfinden und wer diese unterrichtet, sind unvorhergesehene Ausfälle von Lehrkräften, die den Plan durchkreuzen, nicht ungewöhnlich. Leider können Stundenpläne, selbst wenn sie im Allgemeinen gut designed sind, qualitativ versagen, sobald es zu Vertretungsplänen kommt. Derartige Suboptimalität könnte potentiell vorweggenommen werden, wenn die Erstellung von Vertretungsplänen bei der initialen Konstruktion des Stundenplans mit in Betracht gezogen werden. Das Ziel dieser Arbeit ist, diesen Aspekt der Optimierung von Stundenplänen näher zu untersuchen. Hierbei wird ein stochastisches, ganzzahliges lineares Programm vorgestellt, mit dessen Hilfe experimentell untersucht wird, wie vielversprechend eine solche Optimierung sei.

Contents

1 Introduction

The school bell rings. It is a Wednesday morning, 7.45. Mr. L has just pinned the second version of today's substitution plan to the school's main pinboard near the entrance and a crowd of students has formed in front of it, eager to find out whether they can escape today's history lesson or reassuring they remember the right room for the English exam. He is the person responsible for handling the replacement of lessons that were usually held by teachers absent today. It is not unlikely that the substitution plan is updated one or more times during the next 20 minutes, as teachers might still call in sick.
Such a scenario might sound familiar to most: Anyone who has been at a school larger than a few hundred students and a few dozen teachers has been in a similar situation. Lessons cancelled or replaced because of teachers missing is an unavoidable matter and it is not specific to any type of school. Therefore, when talking about teaching efficiency, the quality of school for the students or the recently highly discussed problem of too few teachers ([Boh23]), taking a closer look at substitution plans is just one step away. What is interesting in this context is how they can be improved and how that can influence these topics positively. But before diving into the technical details, let us see more specifically why this is important and find out what exactly needs and can be improved.

1.1 Cancelled lessons in Bavarian schools

The problem of cancelled lessons has always been there, but it was generally considered an issue that was more or less under control ([Son13]). With the recent COVID-19 pandemic there has been a significant spike in cancelled lessons caused by the full or partial closing of schools ([Gri22]). This incident might have raised the incentive to address the problem specifically in relation to the high amount of dropped lessons during this time, but also in the usual situation before and afterwards, the rate of lessons not taking place as scheduled is higher than most expect. A statistical study of the Bavarian State Ministry for Education and Culture (Bayerisches Staatsministerium für Unterricht und Kultus, [FW22]) shows this rate for different types of schools in Bavaria in year 2021/22, along with the distribution between lessons with and without replacement (see Table 1.1): In secondary schools more than ten percent of lessons do not take place as scheduled. To put that into perspective, it is almost a whole month per year that students do not get their lessons as planned, with approximately one week without replacement.

	GS	MS	FöS	RS	GYM
Lessons as scheduled	92.2%	89.5%	89.5%	87.5%	89.0%
Cancelled lessons with replacement	6.4%	8.0%	8.1%	10.9%	8.5%
Cancelled lessons without replacement	1.4%	2.5%	2.3%	1.6%	2.5%

Tab. 1.1: Numbers taken from [FW22]. Abbreviations: GS: Grundschule (Elementary school), MS: Mittelschule, FöS: Förderschule, RS: Realschule, GYM: Gymnasium, years 5-10

1.2 Potential approaches to the problem

During the pandemic, it was a big topic to minimize the total learning time lost to cancelled lessons, meaning minimizing the rate of lessons cancelled. Outside this scope, however, it is hardly possible to reduce this amount, because the bottleneck is generally the availability of teachers - spontaneous absences coming from illness is not something intentional. So instead of trying to change the number of lessons per year that are cancelled, trying to increase the quality of those irregular lessons could be an approach to the problem. For that, we have to first find a measure for what makes the quality of a lesson. Here is an example: Suppose Mr. M is a maths teacher. When M teaches their regular classes, they do that at an efficiency of 100%, 80% of which being the general knowledge a math teacher usually has of the subject, and 20% being proper preparation, including explanations, written and verbal, and exercises. If M is scheduled to be the replacement for another maths teacher that is absent, we can assume that he could do that lesson still at an 80% efficiency due to his general knowledge of the subject, maybe even more if he has access to the preparations of the missing teacher. However, if M is scheduled to replace an absent french teacher, he most likely does not have the general knowledge of the subject, so even if the absent teacher has prepared exercises for the students to do, the efficiency of the lesson is relatively low. It is, in many cases, probably still higher than it would be if the lesson does not get any replacement, simply because the students do something for school during the lesson. Of course those percentage numbers are arbitrary and highly dependent on the teachers ability to improvise lessons and the existence and quality of the preparations received from the regular teacher, but the options can quite clearly put into relation in terms of which one is best for the quality of the lesson:

(V1) Replacing the missing teacher with another teacher of the same subject

(V2) Replacing the missing teacher with any other teacher

(V3) Dropping the lesson without replacing the missing teacher

With the understanding of what makes the quality of an irregular lesson, we can take a look at how we can improve that quality. In fact, there are two conceptionally different approaches: The more obvious one is to try to find methods to make better substitution plans with more replaced lessons being in the higher quality tiers. This might look

relatively easy to be done with the options listed above, and it probably is, but it can get more complex if other theoretically viable options are included. For example, suppose that M is scheduled as replacement for the french lesson in the very same class they teach maths regularly. Instead of taking the low quality french lesson with M as replacement, we could change the lesson to a maths lesson. Possibly M has not yet fully prepared the next maths lesson, but they can still profit from the general knowledge of the subject or use the extra lesson as revision. In that case, the quality of the lesson would be high in exchange for a full french lesson lost for the missing french teacher. In practice, if teachers know they will be absent enough time beforehand, they sometimes organize lesson swaps personally, but theoretically such a thing could also be done afterwards, if teachers involved agree. So there is room for the optimization of substitution plan construction with such more complex mechanisms if one is committed to use them. As said, in practice these mechanisms are only used rarely and are not organized centrally. The other, not quite as obvious approach is taking another step back in the process of the construction of the substitution plan. Indeed, a substitution plan usually only lists the irregular lessons caused by teacher absence, but one could say that, inherently, the major part of the schedule is based on the standard timetable constructed at the beginning of the year. So, assuming we have an algorithm that produces a substitution plan, we could consider this algorithm at the timetable's construction and optimize the latter so that the actual daily schedules produced by said algorithm average at a higher quality. To illustrate this with a very easy example, take the following timetable for two periods and two classes: In both classes the subject of the first period is maths and the second one is french, with different teachers each. If in class 1 the maths teacher is ill, there are only french teachers that can replace the maths teacher. If you change the timetable around, and this is namely done at the very beginning when the timetable is initially constructed, so that class 1 has maths, then french, and class 2 has french, then maths, if the maths teacher of class 1 is ill, there is another maths teacher available to replace them. The second one of these two approaches is what this work is focused on in greater detail:

Definition 1.1. ***Problem:*** *Construct a timetable in a way that allows better substitution plans to be made from it and examine how much of an improvement can be achieved that way.*

We try to do this by starting from an integer linear program designed to construct timetables without considering substitution plans and extending it so that it includes this factor by applying a stochastic programming approach. Using this extended model, which is described in detail in Section 4.1, with synthetic data, we examine how much improvement we can get from this approach (Section 6.1), considering not only the resulting substitution plans, but also other factors that make a good timetable as well as realistic computation times.

2 Automatic construction of timetables

The construction of substitution plans is a relatively easy problem, which can be done by hand in realisic time using a greedy algorithm and produces a solution good enough for nobody to complain. To have a person organize the substitution plan seems to be worth the effort, especially when it comes to more complex changes in schedule, teachers making agreements on their own or changes that are requested spontaneously during the day.
For timetables, this is totally different. Not only is it a major project to find a schedule that fulfills every constraint needed for a valid timetable, it is also very hard to keep the focus and one probably ends up very far away from an optimal solution. For that reason, automatic timetable construction has been quite a relevant topic basically ever since the 60s when computers started to appear more often ([SS80]). Thus, it makes sense to take a look at the current progress in this topic.
When it comes to the construction of timetables, two major categories of methods have surfaced over the decades: metaheuristic approaches ([BCRT15]) and ILP-based approaches. For this work, we want to focus on the latter. The special case of the timetabling problem that the timetable construction at schools can be viewed as has been established under the Curriculum-based Timetabling Problem, CB-CTT for short ([DGMS07]). ILP approaches have been published for this particular problem, most notably there exists a very general ILP formulation of the problem by Burke et al. ([BMPR10]) which is a good starting point for the extension with optimization towards better substitution plans. It is build from three major blocks, which we all need later in some form:

- The most important set of binary variables is an assignment between timeslots, courses and rooms x_{chr}, which is equal to 1 if and only if course c takes place at timeslot h in room r
- From there, we can impose the constraints on this set of variables that are necessary to get a valid timetable: Each course must be scheduled, no more than one course can be in the same room at the same time and a course can only be in one room at the same time. Constraints that model students or teachers only being able to attend one course at a time are also scheduled, but due to the way the ILP is designed, these require a separate set of students and teachers respectively which can be iterated over. Notably, students are also binned into classes, called curricula, hence the name Curriculum-based Course Timetabling.
- Lastly, there has to be an objective function which determines how good or bad a certain solution is compared to others. In this case, a penalty-based system is used,

which counts the number of violations of soft constraints, meaning these do not have to be true, but should be for the most part. An example for a soft constraint we need later on when presenting the whole model in Section 4.1 is that if a course has multiple weekly sessions, these should be spread over the week instead of taking place on one day. The higher the objective value, the more undesired properties in the solution generate penalties, the worse the solution is, so the objective function has to be minimized.

Below is a very brief, more formal overview of the described ILP model. Note that the objective function has been omitted for simplicity in this summary because modelling soft constraints often requires a considerable amount of extra variables, sets and constraints. The function is based on penalties received for certain unwanted structural properties, therefore it has a minimization problem. Generally we only need the basic ideas of this model that we want to extend, and there are some aspects we can leave out.

Definition 2.1. *ILP-model after Burke et al. ([BMPR10]):*

- H = Set of timeslots
- R = Set of rooms
- C = Set of courses
- T = Set of Teachers
- Q = Set of curricula
- lct_c = number of required weekly timeslots for course c
- $x_{chr} = \begin{cases} 1, & \text{if course } c \in C \text{ is scheduled at timeslot } h \in H \text{ in room } r \in R \\ 0, & \text{otherwise} \end{cases}$

Each courses scheduled required amount: $$\sum_{h \in H} \sum_{r \in R} x_{chr} = lct_c \quad \forall c \in C \tag{2.1}$$

Only one course in one room at a time: $$\sum_{c \in C} x_{chr} \leq 1 \quad \forall h \in H, r \in R \tag{2.2}$$

Only one room per teacher at a time: $$\sum_{r \in R} x_{chr} \leq 1 \quad \forall c \in C, h \in H \tag{2.3}$$

Only one course per teacher at a time: $$\sum_{r \in R} \sum_{c \in t} x_{chr} \leq 1 \quad \forall h \in H, t \in T \tag{2.4}$$

Only one course per student at a time: $$\sum_{r \in R} \sum_{c \in q} x_{chr} \leq 1 \quad \forall h \in H, q \in Q \tag{2.5}$$

Expressing the soft constraints used in the objective function requires additional variables, thus the objective function has been omitted for clarity. As it is based on penalties

received for certain unwanted structural properties, the objective value has to be minimized. For more information see [BMPR10] and [BCRT15].
The CB-CTT as a specialization of the generic timetabling problem is an attempt to formalize the general problem of assigning courses to rooms and timeslots and whatever other resources have to be assigned as well. However, as the scope of this attempt is rather large, comprising not only most types of schools in lots of different countries, but also college and university institutions, it is not very surprising that specialization and adaptation to the specific problem is necessary. The formulation of the curriculum-based timetabling problem, as it was made for the International Timetabling Competition 2007 ([DGMS07]), for example, is oriented after the systems used in Italian universities (with example data from the university of Udine) and cannot be ported over to Bavarian secondary schools without respecifying the context. More on the specific modifications necessary are described later on in Section 4.1. Nonetheless, the model shown above is a good example of what the core of an ILP for timetable construction looks like.
At this point, it is also worth to point out the most important advantages and disadvantages of using such a model. The most important positive of ILPs is that they, in theory, produce guaranteed optimal results. This comes, unfortunately, at the cost of very steep runtime inclines, making practical scenarios unsolvable within a realistic time limit. For example, the model mentioned above by Burke et al. ([BMPR10]) cannot be solved for non-trivial instances even if run for several days of computing time (see [BCRT15]). This is a problem to keep in mind when modifying the model and looking for ways to trade optimality for lower runtime is worth the effort.
Even though starting from the above mentioned core and extending it step by step by adding new variables and constraints already sounds like a good idea, we still lack a very important part for our model that needs special treatment. When we want to optimize towards substitution plans, we need to find a way to rate those, and for that, it seems that there is no way around creating them first. However, for creating substitution plans, we need to know about two things we do not have information on yet: The default timetable and which teachers are absent for a certain day. The latter of these problems we can somewhat easily tackle by looking at many cases, creating a substitution plan for each of them and approximate by calculating the expected value over all cases. For the default timetable, this is a bit more subtle. It is not quite promising to first create a timetable, then evaluate how good substitution plans are for it and then try to find a way to improve the timetable in the next iteration. Fortunately, our timetabling optimization problem is not the first one to appear with unknown parameters. In the following chapter, we explore a way to integrate these uncertainties into the ILP while optimizing for their expected value: Stochastic programming.

3 Stochastic programming

3.1 Optimization under uncertainty

The field of optimization under uncertainty is, generally speaking, a generic term for many different methods to deal with parameters that are unknown at the time of calculation. The aim of the field is to find optimal solutions despite the uncertain parts of the data. What makes a solution optimal or generally better than a different one is a matter of definition. On one side of the spectrum is optimization so that the worst possible outcome is as good as possible. This is not only necessary or useful in situations where a worst case scenario can lead to severely negative outcomes, but also when the probabilities of the outcomes of uncertain parameters are unknown and hard to estimate. This type of optimization goes by the term of *robust optimization* ([GH09]). Although dealing with uncertainty, as there are no probabilities to calculate with, robust formulations are naturally deterministic. This is in contrast to *stochastic optimization*, which is useful if such a probability distribution is known or can be estimated. In those cases, optimization is usually done for the expected value. Either of the two approaches can be combined with methods of linear (or non-linear) programming. The modelling of stochastic optimization problems with the help of these mathematical tools is referred to as *stochastic programming* ([BL11]).

3.2 Modelling probabilistic uncertainty with scenarios

A common tool for modelling uncertainty in stochastic programs are scenarios. These are the representation of possible values the uncertain variable might take in the future ([Roc01]). Each scenario has a particular probability to occur, which is in line with the probability distribution of the uncertain variable. Using these scenarios and the corresponding probabilities, a mathematical model can be constructed that considers all the scenarios and optimizes for the best expected value. This method is useful in practice, as long as instances stay small enough to be solved in realistic time. As soon as instances grow in size, especially through increasing scenario set sizes, the method is no longer practically feasible. This is particularly the case if these sets get infinitely big because the modelled variable is continuous. A solution to this is using samples of the set of all possible scenarios that are representative for the probability distribution. These samples are often generated randomly using the given probability distribution and can be adapted in size as desired. Of course, small samples hold the risk of being unrepresentative and unstable and therefore might yield bad results for the whole set of scenarios, big sets, on the other hand, increase the model runtime.

3.3 Applying stochastic programming to timetabling

Typical practical problems where stochastic programming or generally optimization under uncertainty are potentially useful include for example financial planning, inventory and storage management or facility allocation ([Roc01]). An easy example to specifically showcase how uncertainty can surface is described by the farmer's problem, where a farmer aims to sow their crops in a way they result in the best profit without knowing the yields of the harvest and the prices of the crops at the time of the harvest yet([BL11]). In that perspective, as timetabling problems are quite not typical for optimization under uncertainty, primarily because most timetabling problems do not contain any unknown variables at all, it is important to understand how we can apply the theory to the problem. The uncertain variable which we add to the typical timetabling problem is the set of absent teachers on a certain day, so we define a scenario for each possible one. The set of all possible scenarios is therefore the powerset of the set of teachers and gets exponentially big with respect to the size of the latter. This is why we make use of random samples of the full scenario set in the optimization process as well, which is described in more detail in Section 4.4.

3.3.1 Two-stage modelling

If we look at our version of the timetabling problem (Definition 1.1), we can see there are essentially two steps that the model can be divided into in terms of when we have to make certain decisions that fundamentally influence the quality of the solution. The first point is when we construct the timetable. However, the default timetable can be perfect and we do not get a good result if the substitution plans we produce from it mess up the score. Therefore, the second important point is when the information of which teachers are absent becomes known and we have to construct the best substitution plan, ending up at the schedule that happens in reality.
Modelling this sequence of decisions where one is made before and the other one after disclosure of the random variable is referred to as a two-stage program ([BL11], chapter 2.3): The first stage refers to the part of the model which is not dependent on the random variable, whereas the second stage contains the other part which is dependent on the outcome of the uncertain data. Or, to formulate this even more specific to our defined problem, stage 1 contains all factors that make a good timetable without considering teacher absence and stage 2 considers only the influence of the specific substitution plan that we can only create after the set of absent teachers becomes known. This concept of multiple stages, however, is not to be confused with an approach that is run in multiple steps in the sense of multiple iterations. It is still only a single model that is run, where both stages are contained in the object function and the decision practically seem to happen at the same time, in theory. We say in theory because we are going to use two iterations later on despite, but these do not come from the attempt to separate the stages into divided models.

3.3.2 Advantages of using stochastic optimization

In Section 3.1 we have already outlined the primary difference between stochastic and robust optimization. Here we want to quickly sum up why stochastic optimization is the better method to choose for this model: First of all, we can estimate a probability distribution by assuming that each teacher is absent independently and with the same probability, which allows us to calculate the expected value in the first place. Secondly, we have many occurrences of scenarios in practice, meaning that every new school day, we have a new set of absent teachers. Over the course of a year, we get enough cases so that we can assume we approach the expected value overall. Thirdly, really bad scenarios, for example ones where more than half the teachers are absent, happen with a very low probability, so it doesn't make sense to give them a big weight in the optimization process. Moreover, if such particularly unfavorable scenarios actually occur, they are reacted on with mechanisms outside the scope of substitution plans in reality, including for example giving the students a day off. Last but not least, making use of samples instead of the full set of possible scenarios is only sensible when optimizing for the expected value. These arguments quite clearly show that, in our case, stochastic optimization is a quite useful method to apply.

3.4 Application on the problem

Although already briefly mentioned in each of the respective sections, let us sum up the methods useful for our problem (see Definition 1.1). These are the general steps to arrive at a model including both timetable and substitution plan quality in its optimization:

- Create the core of the model with a decision variable x that represents the timetable along with constraints that ensure that x is a valid timetable. The core should be similar to the model by Burke et al. ([BMPR10]), mentioned in Definition 2.1, with the modifications we make to better fit the model to the scope of the problem, in our case Bavarian Gymnasiums.
- Extend the core so that the first stage is properly represented. For this we have to define what aspects we want to rate our timetable after, i.e. what properties we want the timetable to have or not to have. These aspects then have to be modelled as penalties or bonuses into the first-stage part of the objective function.
- Implement a second stage on top of the first one by introducing a new decision variable v that is dependent on x and a scenario u together with the constraints describing the second stage and the extension of the objective function.
- Implement a mechanism to generate and handle random samples of scenarios which are used for calculation instead of the full set of possible scenarios. This point is not directly part of the model, but rather important for the scenario set parameter that is used in the model. See Section 4.4 for more information of this mechanism.

With this knowledge about both the first and the second stage, we can finally start to build the specific ILP model.

4 Creating the model

In the following chapter, we introduce the specific model used to implement the idea of optimizing timetables for the substitution plans they might produce and to examine how relevant the effects are on the quality of the results. Before coming to the description of the model, however, note that the model is not covering an actual real-world scenario, but is oriented after what a Bavarian Gymnasium could look like in terms of classes, teachers and curricula. All the assumptions made for the model are listed in Section 4.1.3. Moreover, Section 4.1 is not necessarily a formally correct description in mathematical terms (for that, refer to Section 4.2), but rather aims to provide way to more easily be able to understand the model and the decisions that are made for it while familiarizing with the notation.

4.1 Detailed model description

4.1.1 First stage

The first thing to do is to specify what resources there are exactly that we want to assign to each other. In Definition 2.1, we see that the relevant sets are the timeslots, courses and rooms available. In our case, as most schools provide a room for each class individually, there are usually enough rooms available at any time so that mostly we can assign rooms unproblematically after obtaining the full timetable. Therefore we do not need to model rooms in the basic assignment. The assignment between teacher and course is something we want to model instead. The ILP in Definition 2.1 assumes that courses can only be held by one specific teacher, as it is common in university courses, but in schools like the Gymnasium, where multiple teachers exist for one subject, this is not the case. With this in mind, we arrive at the sets: The set of *timeslots* H, the set of *courses* C and the set of *teachers* T. The three-dimensional decision variable x with x_{hct} being equal to 1 if course c is held by teacher t in timeslot h is going to represent the timetable.
A timeslot h in this context is defined by its *day* and its *period*, with each day having six periods in the morning and four additional in case of afternoon lessons. Days are named by their name and periods by their number on that day, for example, 'Monday, period 1' is a valid morning timeslot and 'Monday, period 7' is a valid afternoon timeslot. We also denote $D =$ the set of days.
A course c is identified by its *subject* and a set of *groups* the course is for (e.g. 'Maths, 5A' or 'Physical education, 6A and 6B'). Moreover, a course has a number of weekly periods it has to be scheduled and is either a priority course or not. A group is a set of students that always attend the same lessons. For simplicity, we assume that classes

never have to split up for courses because of e.g. different chosen branches, so groups are equal to classes in our case. Therefore, in the model there are no courses that only a split of a class attends, but there are some courses where multiple classes attend (namely PE courses). We denote G as the set of groups/classes, and a group g is identified by its year and an identification character, so that distinguishing between two classes of the same year is possible (e.g. 6A and 6B are different groups).
Thirdly, a teacher t is identified by a unique name and has a set of subjects they are able to teach. Subjects do not have any relevant attributes other than a name to identify them, and we define S as the set of subjects.
From here, we can define the most important constraints: Those that ensure the integrity of the timetable. Specifically, each course has to be scheduled exactly as often as their number of weekly periods say, a group can only attend one course at a time, similarly a teacher can only hold one course at a time, and each course needs a teacher. The last of these restrictions we make more precise by saying each course needs to have exactly one teacher, that is both for a timeslot and in general. These are the most obvious aspects that cannot be violated in a timetable. What is less visible for the outside eye, but sensible, is defining a minimum number of periods a teacher must hold a lesson at. Even though the ruleset is slightly more complicated in practice, for simplicity, we set a minimum number of am_t weekly lessons for each teacher.
Lastly, we have to make some statements about the distribution of lessons in general, especially when it comes to afternoon school. These specifications vary between schools, as they depend on the policies of the individual Gymnasium, so the ones presented here are just one way these constraints could be set. Most importantly, we want all six morning periods to be scheduled a lesson on any day that has no afternoon school. The timing of the periods with breaks in between allows this to be done without any further concerns, however, when afternoon school is scheduled, there needs to be a longer lunchbreak, which in our case we model as a period with no lesson scheduled. This lunchbreak can be either period 6 or 7, and we allow afternoon lessons only on Monday through Wednesday. Additionally, the number of afternoon lessons is to be minimal, but with at most three periods on a single afternoon, we can easily derive the minimal number of necessary afternoon school days from the total weekly lesson count of a class of a certain year. If afternoon lessons take place, these also have to have no gap in between (other than the lunchbreak) and it cannot be a single lesson.
Now we have all the constraints we need for a functioning first stage if the objective function is neglected for the moment. For that, we first have to think about what factors we want to use to benchmark the quality of the default timetable. In the Curriculum-based Course Timetabling example (Definition 2.1) the objective function was penalty-based, meaning they defined some additional soft constraints that a good timetable does not violate for the most part, but when a solution does violate them, this means a higher objective value because of a higher penalty. The aspects rated there are weekly distribution, curriculum compactness, room capacity and room stability. The latter two do not apply to our model, as we do not really model rooms at all and we just assume, rooms are not much of an issue. Curriculum compactness stands for the fact that lessons do not have free periods in between, which can be an issue for university timetabling,

but is also not quite important for our model, as we already forced the distribution of lessons into a certain shape. Weekly distribution, however, we can use as one benchmark factor as well. It refers to multiple lessons of a single course ideally taking place on as many different days of the week as possible. We want to make a small exception there for some courses: There are bonus points for the schedule for *double lessons*. These are two lessons of the same course taking place back to back without a break in between. In our case, a double lesson can be in periods 1 and 2, in 3 and 4 or in 5 and 6, respectively. For some courses that are supposed to be taught in double lessons as much as possible (e.g. Art or Physical education), every double lessons gives the bonus points, for all courses with at least four weekly lessons only one double lessons gives bonus points and for all other courses double lessons are not encouraged at all. Apart from the weekly distribution, we also want our lessons be balanced in the daily distribution. Namely, as periods 2 through 4 are assumed to be those with the highest attention and concentration of the students ([Spo15]), we want the most important subjects to be scheduled in that window as much as possible. Therefore, the timetable is rewarded bonus points for each priority lesson within this window.
This is all the ideas we need for the first stage of the model. Note that the factors chosen for the objective function are somewhat arbitrary and might be seen as more or less useful by different people. Also, modelling these factors as well as some unrelated constraints in practice requires introducing more variables and constraints for how these variables relate to other ones. This has to do with the restrictions of linear expressions on describing certain relationships and not with the concept of these itself, so the details of these variables are presented in Section 4.1.4.

4.1.2 Second stage

For the second stage, we firstly have to define what exactly is uncertain in our case. As highlighted earlier, we consider this to be the set of teachers that is absent and call this a *scenario* u. As we have to consider multiple scenarios, we get U as the set of scenarios, and our related decision variable is v, with $v_{hctu} = 1$ if and only if teacher t is the replacement in course c and timeslot h in scenario u. However, we have to note two important things here: Firstly, t can be empty in case of no replacement being scheduled for the course. This is represented by a separate *dummy-teacher* t_0 that does not have to fit most of the other constraints induced on regular teachers, like the restriction of being only in one course at a time. Secondly, we have to keep in mind that v only lists the replacements, therefore for any information on courses in the regular timetable we have to access x. The main reason for this is that we can massively reduce the number of variables v_{hctu} if we leave out the lessons that take place as planned. For the theoretical model, this does not really matter, but in practice we need the runtime reduction.
For the variable v, we need to add two constraints to the model: A teacher other than t_0 can still not be in more than one course at a time and every missing teacher has to be replaced by a teacher present or t_0. For the objective function, we do not need any additional advanced mechanisms: We can simply determine, which of the cases in Section 1.2 applies for every v_{hctu} that is 1 and add up the penalties generated by them.

4.1.3 Assumptions for the model

Here we list all the assumptions made for the model. For more context on each bullet point see the corresponding part in the detailed description above (Section 4.1).

- There is no shortage of classrooms of big enough size.
- There exist no courses that a class has to split up for.
- The quality of a timetable can be evaluated from the weekly distribution of courses, the presence and amount of double lessons and good priority course scheduling. For weekly distribution it is desired for a course to take place on as much days as possible rather than stacking on a single day. An exception to this are double lessons, which are encouraged to appear once per week for courses with at least four weekly lessons. Some courses are also generally desired to be taught in double lessons. Lastly, priority courses should be scheduled in periods 2 - 4.
- Any teacher can be chosen as replacement teacher, as long as they are not absent and not busy with a different course at the time. The option of scheduling no replacement in this regard is modelled as scheduling a dummy-Teacher t_0 as replacement, which is a teacher-like object that teaches no subject, is never absent and can be scheduled as teacher of multiple courses at the same time, but only as replacement.

In addition to these assumptions, there is one relevant element of the model that could be considered a parameter choice, but is hardcoded into the model. This element is the format of the day and the week in terms of how periods are distributed. In the model it is assumed that there are ten timeslots a day, six in the morning and four in the afternoon, but afternoon school can only be on Monday, Tuesday and Wednesday. Also, if there is afternoon school, there must be exactly one free period as lunchbreak in either period 6 or 7 of the respective day, and afterwards lessons must form a block without gaps in between. Moreover, there must not be only a single lesson on the afternoon.

4.1.4 More details on the objective function and additional necessary variables

As outlined at the end of the description of the first stage (Section 4.1.1), we need some additional variables to be able to compute the desired penalties and bonuses in the objective function. These variables do not need additional constraints other than linking them to x in some way. Sometimes, two variables are necessary to achieve the desired link that is needed for a proper representation of the corresponding soft constraint. The variables are the following:

- For the soft constraint of weekly distribution we have to count the number of days d a course c is scheduled a lesson. For this, we introduce two variables: The decision variable z_{dc} with z_{dc} is one if and only if course c is scheduled on day d at least once, and wd_c counting the number of days course c is scheduled by adding up z_{dc}

for the respective course. In the objective function, we can take $pen^{WD}(am_c - wd_c)$ to get the penalty for the weekly distribution of each course.

- For the soft constraint of double lessons we also need two variables in a similar fashion: We want the variable dl_c to be one if there is at least one double lesson of course c. For that, in order to be able to count the number of double lessons, first we need a separate variable dlh_{ch_2} recording every single double lesson with dlh_{ch_2} being equal to one if and only if there is a double lesson at the *double timeslot* h_2. h_2 is from the set DH, which is simply the set of pairs of timeslots that two times the same course being scheduled for is considered a double lesson. dl_c then is one, if the sum of such double lessons over all double timeslots is at least one. In the objective function, we can multiply dl_c for each course with the desired bonus bon^{DL}, but we want to do so only if course c has at least four weekly lessons and is not in the set of courses that are supposed to be held in double lessons as much as possible. For such courses, we want to reward every double lesson per week regardless of how many weekly lessons they have, so we simply multiply the sum of dlh_{ch_2} over all double timeslots h_2 by the desired bonus bon^{DL2}. Note that, when we minimize the objective functions, bonus factors have to be negative. Also, we want the scheduling of a double lesson be more attractive then distributing the two lessons on different days, so $|bon^{DL}|$ needs to be greater than $|pen^{W}D|$.

- For the soft constraint of preferring scheduling priority courses at periods 2, 3 or 4, we do not need additional variables, we can simply count the number of priority courses in the specified timeslots and multiply the result by the desired bonus bon^{PC}.

- Not related to the objective function, there is another variable which is necessary for describing the fact that a course should have exactly one teacher, not specific to the timeslot the course is in. It has less to do with the inability of a teachers being at multiple places at once, like the similar, timeslot-specific constraint, but rather describes the sensible claim that two teachers alternating in holding the same course tends to be not very advantageous at all. For describing this relationship of each course having a single distinct teacher, we introduce a variable w_{ct} that is equal to one if course c is held by teacher t. In addition to linking this variable to x, we can easily formulate the just mentioned constraint by setting the number of teachers for a course to exactly one.

- Moreover, for counting the number of afternoon school days of a group, which we need for ensuring this number stays below a predefined threshold, we need a variable a_{dg} that describes whether group g has afternoon school on day d.

4.2 Formulating the Model

In this section, we present the mathematical formulation of the stochastic integer linear program. The model has been implemented with the python library docplex that pro-

vides an interface to CPLEX as solver, but for this section, we focus on the theoretical, mathematical formulation. Therefore, some minor deviations that had to be made due to technical matters with python coding are neglected for now and outlined in more detail afterwards (see Section 4.3).

Sets, constants and parameters:

- T = the set of teachers
- H = the set of timeslots
- C = the set of courses
- $U \subset 2^T$ = the set of scenarios
- $S_t \subset S$ = the subjects of teacher t
- d_h = the day of timeslot h
- p_h = the period of timeslot h
- G_c = the set of groups of course c
- s_c = the subject of course c
- $prio_c \in \{0, 1\}$ = the priority of course c
- $G = \bigcup_{c \in C} G_c$ = the set of groups/classes
- $C_g = \{c \in C \mid g \in G_c\}$ = the set of courses for a group
- $S = \{s_c \mid c \in C\}$ = the set of subjects
- $D = \{d_h \mid h \in H\}$ = the set of days
- $DH = \{(h, i) \mid p_h = 2n - 1, p_i = 2n, n \in \{1, 2, 3\}\}$ = the set of double timeslots
- $H^i = \{h \in H \mid p_h \leq i\}$ = the set of timeslots before or at period i
- $H_j = \{h \in H \mid j \leq p_h\}$ = the set of timeslots at or after period j
- $H_j^i = H^i \cap H_j$ = the set of timeslots with their period within the interval [j, i]
- $H_j^i(d) = \{h \in H_j^i \mid d_h = d\}$ = the set of timeslots on day d with their period within the interval [j, i]
- C_{DL} = Set of courses that is supposed to be taking place in double lessons as much as possible
- $C^i = \{c \in C \mid am_c \leq i\}$ = the set of courses with i or more required weekly periods

- $C_j = \{c \in C \mid j \leq am_c\}$ = the set of courses with j or less required weekly periods
- $C_j^i = C^i \cap C_j$ = the set of courses with required weekly periods in the interval [j, i]
- $T_c = \{t \in T \mid s_c \in S_t\}$ = the set of teachers that can teach course c
- t_0 = a teacher-like object representing that no teacher is scheduled
- am_c = the amount of weekly timeslots course c has to be scheduled
- am_t = the minimum amount of weekly timeslots teacher t has to be scheduled
- af_g = the maximum amount of afternoon school days for group g, this is usually defined by their year
- pen^{WD} = the penalty for low spread of a course with multiple periods over multiple days of the week
- bon^{DL} = the bonus for double lessons for courses not in C_{DL} (only once per course)
- bon^{DL2} = the bonus for double lessons for courses in C_{DL} (multiple times per course)
- bon^{PC} = the bonus for priority courses to be scheduled in periods 2, 3 or 4
- pen^{V1} = the penalty for a substitution lesson with the replacement teacher teaching the subject of the course
- pen^{V2} = the penalty for a substitution lesson with the replacement teacher not teaching the subject of the course
- pen^{V3} = the penalty for a substitution lesson with no replacement teacher

Variables:

- $x_{hct} = \begin{cases} 1, & \text{if teacher } t \in T \text{ is scheduled for course } c \in C \text{ at timeslot } h \in H \\ 0, & \text{otherwise} \end{cases}$
- $v_{hctu} = \begin{cases} 1, & \text{if teacher } t \in (T \setminus u) \cup \{t_0\} \text{ is scheduled as replacement for course } c \in C \\ & \quad \text{at timeslot } h \in H \text{ in scenario } u \in U \\ 0, & \text{otherwise} \end{cases}$
- $w_{ct} = \begin{cases} 1, & \text{if teacher } t \in T \text{ is scheduled for course } c \in C \\ 0, & \text{otherwise} \end{cases}$
- $a_{dg} = \begin{cases} 1, & \text{if group } g \in G \text{ as afternoon school on day } d \in D \\ 0, & \text{otherwise} \end{cases}$

- $z_{dc} = \begin{cases} 1, & \text{if course } c \in C \text{ is scheduled on day } d \in D \\ 0, & \text{otherwise} \end{cases}$
- $wd_c =$ the number of days course c is scheduled
- $dlh_{ch_2} = \begin{cases} 1, & \text{if course } c \in C \text{ is scheduled as double lesson at double timeslot } h_2 \in DH \\ 0, & \text{otherwise} \end{cases}$
- $dl_c = \begin{cases} 1, & \text{if course } c \in C \text{ is scheduled as double lesson at least once} \\ 0, & \text{otherwise} \end{cases}$

Constraints (general integrity of the timetable):

$$\sum_{h \in H, t \in T} x_{hct} = am_c \qquad \forall\ c \in C \tag{4.1}$$

$$\sum_{h \in H, c \in C} x_{hct} \geq am_t \qquad \forall\ t \in T \tag{4.2}$$

$$\sum_{t \in T, c \in C_g} x_{hct} \leq 1 \qquad \forall\ g \in G, h \in H \tag{4.3}$$

$$\sum_{c \in C} x_{hct} \leq 1 \qquad \forall\ h \in H, t \in T \tag{4.4}$$

$$\sum_{t \in T} x_{hct} \leq 1 \qquad \forall\ h \in H, c \in C \tag{4.5}$$

$$\sum_{t \in T} w_{ct} = 1 \qquad \forall\ c \in C \tag{4.6}$$

$$w_{ct} = 0 \qquad \forall\ c \in C, t \in T \setminus T_c \tag{4.7}$$

$$\sum_{h \in H} x_{hct} \geq w_{ct} \qquad \forall\ c \in C, t \in T \tag{4.8}$$

$$x_{hct} \leq w_{ct} \qquad \forall\ c \in C, h \in H, t \in T \tag{4.9}$$

Constraints (afternoon school):

$$\sum_{d \in D} a_{dg} \leq af_g \qquad \forall\ g \in G \tag{4.10}$$

$$\sum_{c \in C_g, t \in T} x_{hct} \geq 1 \qquad \forall\ g \in G, h \in H^5 \tag{4.11}$$

$$\sum_{h \in H_6^7(d), c \in C_g, t \in T} x_{hct} = 1 \qquad \forall\ d \in D, g \in G \tag{4.12}$$

$$\sum_{c \in C_g, t \in T} x_{hct} \geq \sum_{c \in C_g, t \in T} x_{ict} \qquad \forall\ h \in H_8, i \in H_{p_h+1}(d_h), g \in G \tag{4.13}$$

$$\sum_{h \in H_7(d), c \in C_g, t \in T} x_{hct} \geq 2a_{dg} \qquad \forall\ d \in D, g \in G \tag{4.14}$$

$$\sum_{c \in C_g, t \in T} x_{hct} \leq a_{d_h g} \qquad \forall\ h \in H_7, g \in G \tag{4.15}$$

Constraints (stage 1 components):

$$\sum_{t \in T, h \in H(d)} x_{hct} \geq z_{dc} \qquad \forall\ d \in D, c \in C \tag{4.16}$$

$$\sum_{t \in T} x_{hct} \leq z_{d_h c} \qquad \forall\ h \in H, c \in C \tag{4.17}$$

$$\sum_{d \in D} z_{dc} = wd_c \qquad \forall\ c \in C \tag{4.18}$$

$$\sum_{t \in T} \frac{x_{ict} + x_{jct}}{2} \geq dlh_{ch_2} \qquad \forall\ c \in C, h_2 = (i,j) \in DH \tag{4.19}$$

$$\sum_{h_2 \in DH} dlh_{ch_2} \geq dl_c \qquad \forall\ c \in C \tag{4.20}$$

$$dlh_{ch_2} \leq dl_c \qquad \forall\ c \in C, h_2 \in DH \tag{4.21}$$

Constraints (stage 2 components):

$$\sum_{c \in C} (x_{hct} + v_{hctu}) \leq 1 \qquad \forall\ h \in H, t \in T \setminus u, u \in U \tag{4.22}$$

$$\sum_{t \in u} x_{hct} = \sum_{t' \in (T \setminus u) \cup \{t_0\}} v_{hctu} \qquad \forall\ h \in H, c \in C, u \in U \tag{4.23}$$

Objective function components (stage 1):

$$f_{WD} = \sum_{c \in C \setminus C_{DL}} (am_c - wd_c) \tag{4.24}$$

$$f_{DL} = \sum_{c \in C_4 \setminus C_{DL}} dl_c \tag{4.25}$$

$$f_{DL2} = \sum_{c \in C_{DL}, h_2 \in DH} dlh_{ch_2} \tag{4.26}$$

$$f_{PC} = \sum_{h \in H_2^4, c \in C, t \in T} prio_c x_{hct} \tag{4.27}$$

$$f_1 = pen^{WD} f_{WD} + bon^{DL} f_{DL} + bon^{DL2} f_{DL2} + bon^{PC} f_{PC} \tag{4.28}$$

Objective function components (stage 2):

$$f_{V1} = \sum_{h \in H, c \in C, t \in T_c \setminus u, u \in U} v_{hctu} \tag{4.29}$$

$$f_{V2} = \sum_{h \in H, c \in C, t \in T \setminus (u \cup T_c), u \in U} v_{hctu} \tag{4.30}$$

$$f_{V3} = \sum_{h \in H, c \in C, u \in U} v_{hct_0 u} \tag{4.31}$$

$$f_2 = \frac{pen^{V1} f_{V1} + pen^{V2} f_{V2} + pen^{V3} f_{V3}}{|U|} \tag{4.32}$$

Note: In this model, we assume U is a random sample of scenarios obtained in the way described below in Section 4.4. If U was simply the set of all to be computed scenarios, the objective function for the second stage needs to multiply each variable v_{hctu} with the probability of scenario u $P(u)$ and the corresponding penalty pen_{ct}, resulting in a function like

$$f_2 = \sum_{h \in H, c \in C, t \in T, u \in U} pen_{ct} v_{hctu} P(u)$$

Objective function:

$$\text{minimize } W f_1 + (1 - W) f_2$$

where $W \in [0, 1]$ is the percentage weight for the first stage.

The constraints and functions represent the following statements:

- (4.1) ensures each course is scheduled exactly enough weekly timeslots.
- (4.2) ensures that the minimum required weekly lessons each teacher must hold are met.
- (4.3) and (4.4) limit the amount of parallel courses a group or teacher attends to a maximum of 1.
- (4.5) limits the amount of teachers simultaneously holding the same lesson to a maximum of 1.
- (4.6) fixes the relation of each course generally having exactly one responsible teacher.
- (4.7) ensures that in the default timetable the teacher assigned to a course is able to teach the subject of the course.
- (4.8) and (4.9) link the variables x and w together, so that each x_{hct} does only use c-t assignments that are permitted through w_{ct} and vice-versa.
- (4.10) through (4.14) constrain the distribution of lessons over the timeslots, most notably the format in which afternoon school is allowed. More specifically, (4.10) limits the maximum amount of afternoon days to not exceed a given number, (4.11) forces the morning lessons (from period 1 to 5) to never be empty. (4.12) implies that there must be exactly one period of lunchbreak, either period 6 or period 7 of each day. This also means that days without afternoon school forcibly have the 6th period of the day scheduled a lesson. (4.13) and (4.14) state that afternoon school must take place in one block of at least two lessons without interruptions, meaning there is no free gap between two periods and the first period of the block

is in period 7 or 8, depending on whether the lunchbreak period is in period 6 or 7, respectively. While (4.14) already prohibits a_{dg} to be 1 if there are no lessons on the afternoon of day d for group g, (4.15) in the contrary does ensure that it is 1 if there are lessons.

- (4.16) and (4.17) link the variables x and z together, so that z makes a statement about whether a course is scheduled on certain day. With the help of this variable z, (4.18) counts on how many days the course is scheduled each week.
- (4.19) defines the relation of x to dlh so that dlh is only 1 if there is a double lesson at the specific double timeslot. dlh can be zero from this constraint alone even if there exists a double lesson at the respective double timeslot, however, in the objective function, double lessons are only rewarded bonus points, encouraging $dlh = 1$ whenever possible.
- (4.20) and (4.21) link the variable dl to dlh, so that dl states whether there is a double lesson for course c at any double timeslot.
- (4.22) extends the restriction of a teacher being able to attend only one course at a time to also include substitution periods. A teacher can neither be scheduled as a replacement teacher if they have a regular lesson at the respective timeslot nor can they be scheduled to replace teachers in two different courses at the same time.
- (4.23) ensures that each missing teacher is actually scheduled a replacement for. Including the dummy-teacher t_0 in the set of potential replacement teachers enables the option of no replacement being scheduled without violating this constraint.
- (4.24) counts the number of times a course repeats on a day. Expressed differently, it counts the number of days the distribution is short of the maximal potential number of days, which would be reached if each course is only scheduled once per day.
- (4.25) and (4.26) count the number of double lessons that are worth bonus points to be rewarded. The former counts one for each first double lesson of a course not in C_{DL} with at least four weekly periods. If the course has more than one double lesson, it still counts only once. On the contrary, (4.26) counts one for every unique double lesson, but only for courses in C_{DL}, as these are desired to take place in double lessons as much as possible.
- (4.27) counts the number of times a priority course is scheduled in periods 2, 3 or 4.
- (4.28) combines the different counters of the first stage, multiplying them with the corresponding penalty or bonus weights.
- (4.29), (4.30) and (4.31) count how often each of the cases mentioned in Section 1.2 appears.

- (4.32) combines these counters of the second stage, multiplying them with the corresponding penalties and finding the expected value of the results over all scenarios u in the sample set U.

4.3 Deviations of the implementation

For practical reasons, some parts of this model have been modified in the implementation. Most notably, the space of the variables has been reduced to smaller sizes whenever possible, to reduce runtime and memory when solving the model. In (4.7), a major part of variable w_{ct} and derived from that x_{hct} is set to 0, more specifically all w_{ct} and x_{hct} where $s_c \notin S_t$. By using replacing either $t \in T$ with $t \in T_c$ or, the other way round, $c \in C$ with $c \in C_t$, with $C_t = \{c \in C \mid s_c \in S_t\}$, in the creation of the variables x and w and in every involving constraint, we can omit (4.7) as the variables are implicitly not defined anymore. Note that replacing these sets can not always be done arbitrarily.

4.4 Generation of scenario samples

Lastly, a thing that we must not forget is the generation of the scenario set U. As already hinted at multiple times, we cannot compute an arbitrarily high number of scenarios, certainly not all of the possible ones, so we need to generate a random sample of scenarios that we can use in the model. We do this in the following way: Assuming that the probability of teachers being absent is $prob = 0.1$, and that the absence of each teacher is independent to each other, we can determine for every teacher randomly, whether they are absent or not, hence creating a single scenario u. Repeating this multiple times gives us the random sample of scenarios U.

5 Evaluation with synthetic data

This chapter describes the experiments that are conducted with help of the model. In general, the model is oriented after the system of Bavarian Gymnasiums, and even though no data are directly collected from those, in the authors opinion they are plausible to appear in reality. The focus of the experiment does not lie in generating timetables viable for practical use, but rather the exploration of how promising the approach of optimizing for substitution plans really is. More on why the experiment is still valuable despite that is discussed in Section 6.1.

5.1 Basis of the input data

Before taking a look at the results of the model, it is necessary to understand the input data. In our case, the important sets are the timeslots H, the courses C, the teachers T and the scenarios U. The set of timeslots includes ten periods on Monday, Tuesday and Wednesday and six periods on Thursday and Friday. For the set of courses C, it is more or less fixed by legislation how many weekly lessons courses of each subject have to feature each year, with differences e.g. between the six branches or different choices of foreign languages. Thus, there are many different ways how branches and foreign languages could be combined, and a Gymnasium usually only features a few of those. For the input, courses for years 5 through 10 are defined, with English being obligatory first foreign language, second foreign language split between French and Latin and branches split between the third foreign language Spanish or science as additional focus, which is one of the more common combinations of branches and languages schools offer. As mentioned, the model does not support the splitting of classes for simplicity, so one could view this as 'all students chose their combinations in a way so that the classes could be formed without the need of splitting', e.g. there is no class containing both Latin and French students. Physical education courses are often organized as combined classes, but split sex; in this case, these courses are not split by sex, but are still modelled as combined courses.
The number of classes per year is chosen to be four, which allows for a good coverage of different subject combinations and is also a common average on Gymnasiums. The reason why only for years up to year ten courses are generated is for one that for the nine-year curriculum introduced in Bavaria in year 2018/2019, which has been faded in by only being applied to new classes, year eleven has not yet appeared in this new system. Secondly, qualification stage, which have been years 11 and 12 in the past curriculum and are years 12 and 13 in the new one, have a slightly different system no longer being oriented by branches, and therefore, again for simplicity, is not modelled.
For the set of teachers T, a number of teachers with plausible subject combinations is

handpicked so that each subject has enough different teachers while keeping the total number of teachers so that the minimum weekly lessons of them is only slightly exceeded. Lastly, the set of scenarios is generated in the way already described at the end of Section 4.1.2: Each teacher is assumed to be absent with a certain probability, and for that a scenario as a random sample of absent teachers is generated. This is done multiple times, until the desired number of scenarios is reached. The probability is assumed to be 0.1 for all teachers as this is approximately the rate of lessons not taking place by schedule, as shown in Table 1.1.
For the parameters that have to be specified, most of them do not change over multiple instances executed. am_c is set as attribute of the courses and defined when the courses are, chosen as the values given by the curriculum. af_g can be derived from am_c, as it is the smallest number of afternoon school days possible without violating any of the other constraints. am_t is chosen as 21 for all teachers, which is slightly below the average value in reality. Lastly, the bonuses and penalties which are essentially the weights of different aspects in the objective functions can be chosen somewhat arbitrarily, depending on how important one sees each of the aspects. The bonuses should be negative numbers, as the ILP minimizes the objective function, while penalties should be positive. pen^{V1} should be the smallest of the second stage weights, as it is the most favorable case of replacement. The weight between the two stages is represented separately by another weight W. For our experiment specifically, we choose the bonus and penalty parameters as follows:

- $pen^{WD} = 3$
- $bon^{DL} = -5$
- $bon^{DL2} = -10$
- $bon^{PC} = 2$
- $pen^{V1} = 0$
- $pen^{V2} = 3$
- $pen^{V3} = 5$

5.2 First experiment: Practical runtime

The full data set of the model consists of 42 Timeslots, 33 teachers and 280 courses spread over 4 classes per year and 6 years. The scenarios are not fixed and generated randomly at the beginning of the calculation. Running the model with the full data and all mentioned aspects already shows that, despite the effort of reducing the variable count as much as possible, we remain at 56826 variables of x, 7633 variables important for stage 1 objective function calculations and approximately between 300000 and 400000 variables of v per scenario (the exact number depends on the scenarios). Here we run into problematically high runtimes even with only one non-trivial scenario in the set

U. This is, of course, not quite a positive outcome, but it is also not a too surprising one. Either way, a change in strategy is necessary if we want to experimentally examine whether the effort of optimizing for substitution plan quality is worth it.

5.3 Experiment strategy

Going forward, the following strategy is pursued: We run the model with the full data set with just the first stage to obtain a *base solution*. Even with only the variables of x and stage 1, the model will most likely not terminate within a realistic time frame either, but we can abort the calculation at any time and use the best solution found at that point. In our case, the model was run for about 10.5 hours. On top of that comes the time the model needs for the creation of its variables, constraints and objective function components and for preprocessing, which we refer to as *loading time* of the model. Note that, in this case, this time is not measured, but is found not to be significantly long. The solution obtained from the model after the mentioned time gives us the base solution we use for further optimization. It has a gap of 28.75 and an objective value of -38, but these numbers do not say very much on their own apart from the gap telling us that the solution probably is not very close to being optimal.
Starting from the obtained base solution, we can create smaller instances by isolating a number of classes we want to rerun the model for while keeping the base timetable for all other groups. We do this not only for different reoptimization group sizes, but also for different sizes of scenario sets and for different weightings W of the two stages relatively to each other. To get representative results, we repeat the experiments multiple times for each combination of reoptimization group size, scenario set size and objective function weighting. Our first objective with this is to find out how realistic it is to expect the rerun to finish calculation within a practical time or at least return an output with small gap. Afterwards, we can take additional steps to evaluate the results further.

5.4 Analysing the runtime of instances

To find out whether a second iteration finds a solution within a realistic time frame, a time limit of 3600s is set for the model. This limit does not include the loading time of the model. We run the second iteration for only stage 1 (W equals 1), only stage 2 (W equals 0) and the stages combined with equal weight (W equals 0.5). We do this once for each class, for some random combinations of two classes and for each year. For stage 1, we do not need any scenarios as they do not affect stage 1 anyways, but for stage 2 and for the combined cases we create scenario test sets of small, medium and large size consisting of 5, 15 and 30 scenarios, respectively.
For stage 1, models for group sets of one or two classes at a time solve unproblematically, with runtimes averaging at a few seconds (see Table 9.2. Increasing the group size to one year comprised of four classes, we can see that the runtime rises rapidly with only four of the six years solving in time. Interestingly enough, the average runtime is still not too close to the time limit with only about 25 minutes, which could be explained by

the fact that later years are much more complex to solve due to a significantly higher number of subjects. Moreover, the gap for the cases that remained unsolved within the time limit stayed relatively low with an average of 0.035.
For stage 2, we see a totally different picture (see Figure 8.1. Even though the trend of higher group size leading to higher runtime is still prevalent, the model runtimes overall stay low, with only the one-year group size for 15 scenarios breaching the one-minute average. However, we can observe a different time consuming factor rise to relevance: The loading time exceeds the two-minute mark already for just five scenarios and two classes, and for 30 scenarios it takes well above two hours to load the model alone. If we included the loading time in the time limit, these runs would terminate without a solution found. For combinations with very high loading times only a single instance has been run, so the presented data point might not be representative for the actual average of the category.
If we combine the two stages with equal weight, we might expect that the two runtime-increasing factors of the categories with only one stage considered combine as well. In the combined stages (see Figure 8.2), the runtime of the models themselves is higher and heavily dependent on the scenario set size. The loading time of the model, however, is very similar to what we observed in stage 2 experiments.

5.5 Analyzing the stability of solutions

As we have seen, we get quite different values for model runtime and loading time when we change the models' group size and scenario set size. As scenario sets we generate a random sample of the possible scenarios U and have the timetable optimized based on that. The main purpose of this method is, as we have explored in Section 4.4, the reduction of the size of U to something we can handle. In runtime and loading time, this simplification has indeed been reflected: Models with smaller scenario sets take shorter to load and solve. However, we cannot reduce this size as much as we want without expecting a negative impact on the result of the calculation. The problem is that, if the scenario set is too small, the timetable might be good for some of the possible scenarios, certainly for those contained in the set, but it might be bad for others.
For this reason, we examine in this section, how the amount of scenarios used in the optimization process affects the stability of the result, i.e. how many scenarios are necessary to get a small deviation from the expected value. We do this by taking a number of second iteration results and reevaluate the whole timetable for two sets of scenarios. One of these is the set of scenarios used for optimization during the second iteration and the other one is a randomly generated sample of scenarios containing 100 cases, which should be a good approximation for the expected value. This reevaluation means essentially the creation of the optimal substitution plan for each scenario. As the options for taking reaction on teachers being absent in our case do not include the shifting of timeslots, but only choosing the best suited replacement teacher, we can generate such an optimal substitution plan for a single timeslot at a time, using a small ILP which is similar to the second stage of our main model, with $|H|$ being equal to one. That way,

we can iterate over all timeslots and add up the results to obtain an optimal substitution plan for the whole timetable. For the mentioned stability experiment, we are primarily interested in the combined stages case, and we take results from the two-classes category, as those seem to be of the largest group size while still quite reliably staying at not too high runtimes.
In Figure 8.3 we can see the relative deviation for some instances of the scenario set used during optimization to the the corresponding 100-scenarios evaluation. To get a feeling for how the deviation is for very large sets, the graph also shows how far from the 100-scenario evaluation a random 50-scenario subset deviates. Moreover, in Figure 8.4 we can see a comparison for the absolute values for both the average amount of weekly cases of same-subject replacement and not-same-subject replacements for the same timetable results. As the penalty pen^{V1} for the former has been chosen to be zero, the overall second stage objective value is proportional to the not-same-subject replacements.
We can see that the deviation is considerably smaller for 15 scenarios than for 5 scenarios. However, there are still significant outliers, and compared to how the 50-scenario subset performs, all values are quite high. Considering the fact that even the 50-scenario subsets show a visible spread in the deviations, we cannot expect that the stability rapidly increases between 15 and 50 scenarios. It might be possible to go far above 15 scenarios and, for example, run more instances with 30 scenarios to obtain more stable results, that is if the runtime (or more specifically the loading time) is an issue of lesser degree. To sum this up, for the following quality analysis, we simply have to keep in mind that deviations exist on a level that can cause fluctuations in the resulting data.

5.6 Analyzing the quality of solutions

Finally, of course we want to find out whether the calculations that include substitution plan quality actually improve the timetable. Quality in this context means that the objective value is as small as possible and therefore includes all aspects of first and second stage. For this evaluation, we simulate ten weeks of school where on each day a new scenario occurs, so the total amount of scenarios considered is 50. For each timetable, we run this simulation, generating the optimal substitution plan the same way we did for the stability experiment (described in Section 5.5), except we evaluate each scenario for only one day instead of the full week.

5.6.1 First stage

If we take a look at the second iterations that only optimized for stage 1, i.e. substitution plans are not at all considered in these solutions, we can observe that indeed there has been a clear improvement compared to the base solution (Table 9.5). It confirms that the base solution that we obtained with a gap of 28.75 was indeed far from optimal. We also have to keep in mind that in the second iteration we have only made changes in a small part of the timetable, depending on the chosen group size. This is the reason for why optimizing for two classes results in about double the improvement of optimizing

for a single class, even though this scaling is probably not linear when going for higher group sizes.

5.6.2 Second stage

For the second stage, we can also find visible improvements in the second iteration runs. That is, if we assume the averages to be representative. What is different to the stage 1, however, is that, for some instances, the objective value has actually become worse. As the graph in Figure 8.5 shows, this is especially the case for low scenario set sizes, but not limited to. In fact, every category with multiple instances evaluated has their maximum objective value higher than what the base solution resulted in.
Responsible for these fluctuations is likely the limited stability due to small scenario sets, but as we have also seen in Section 5.5, even for larger sets the results have relatively high error potential with dependence on the scenarios in the set.
In this context, it is also hard to say whether the objective value of the base solution is representative for the average, as we have only a single value available. In either way, the results for the smallest scenario set size of 5 are rather close to the base solution, especially in comparison to their own error. For the other categories, we can find that, despite the fluctuations, the quality of replacement choices has clearly improved. In particular, this quality manifests through less lessons being replaced by a teacher not of the same subject, which is directly proportional to the objective value of stage 2 in our case. Similar to what we already mentioned for stage 1, we have to keep in mind that we have made changes only in parts of the timetable, so scaling this to all classes might result in much more significant improvements.

5.6.3 First and second stage combined

If we combine these two findings of both first and second stage, we could expect that the phenomena seen on the two objective values partially transfer to the results of the combined objective function. This would mean that a major part of the improvement comes from the first stage, as in Table 9.5 we have seen average improvements between -38 and almost -170, depending on the group size. On the contrary, second stage improvements have remained at values around -5.
If we take a look at Table 9.6, specifically at the section for combined stages, what we observe is, firstly, that the first stage objective value of the combined stages is pretty close to the result of the first stage value obtained by the stage 1 experiment. In fact, many solutions, especially those with small scenario sets share the exact same stage 1 objective value even in terms of its components with the corresponding stage 1 iteration results.
Despite that, the average total objective value of the combined-stages runs are always lower than the objective value of the stage 1 runs (if the model has been solved in time). As the results of optimizing for stage 1 in the stage 1 runs impose a lower bound on the first stage component of the combined stages runs, this improvement can only be caused by the second stage. (In the table, -104 in stages-combined/two classes/30 is

lower than the value in stage-1/one-year because the former is a single instance whereas the latter is an average over multiple different ones.) The fact that improvements on the total objective value have been made consistently through all cases also suggests that this does indeed not come from random fluctuation as well. Thus, even though the weight of the second stage has much less contribution to the absolute improvement in the objective value than the first stage, it clearly has an impact on the quality of the timetable as a whole.

6 Discussion

Lastly, in this section we want to discuss how the results of the experiment could be interpreted, what we could learn from the experiment in general and suggestions on how to potentially continue research in this topic. Note that, even though some statements might be based on data available through the experiment or other sources, this chapter generally is a reflection of the authors personal opinion.

6.1 Significance of results

The first and foremost question that has not been answered yet is: Is the idea covered in this topic, namely the one of optimizing timetables on schools, useful in practice? Unfortunately, giving a definitive answer on this at this point is probably a bit too early. The experiment conducted does show that improvements can be made. How relevant these improvements are, however, is still dependent on many factors not explored in this experiment. This does not only include the fact that with limited time and computing power the possibilities to conduct the experiment were limited, for example when it comes to running more complex models for longer than a few hours. It is also the case that the model itself is a simplification of the systems used in practice. Thus, it is assumed, but not guaranteed that if the model is extended so it better suits these systems, there are no substantial deviations in the results from those generated by the model used in this experiment.
The main purpose of the experiment conducted is to find out whether it is worth to even consider including substitution plans as aspect to optimize timetables for. It can be seen as a starting point and in that context it can be considered a success as, after all, the results indeed show potential for improvement. To put this into a more meaningful perspective: Even though the data is not quite clear about this value, let us assume that per week we have one more replacement by a teacher of the same subject as the course with a optimized timetable than with one without. This corresponds to 3 penalty points that are saved in the second stage objective value, so looking at the experiment, we see this is achievable by runs that have been optimized with two classes. Now these runs have only been optimized with two classes, so if we repeat this process multiple times, we can probably increase the effect by a multiple. (We have to note, however, that the effect increase probably shrinks with multiple iterations).
So let us assume in the end we can turn five replacements from ”not the same subject” to ”the same subject” with such a method. In the scenario set used for the ten-weeks simulation in Section 5.6, the average number of total weekly lessons that a replacement is needed for is 66.6, with 44.7 replaced by teachers of the same subject and 21.9 replaced by teachers not of the same subject (and 0.0 lessons with no replacement). If

we apply the effect assumed above, we would get 49.7 same-subject replacements, which is approximately 75% of all replacements instead of only 67%. Of course, the numbers picked in this example are optimistic, to showcase the potential continuing research in this topic might have.

6.2 Potential further research

So how specifically can this topic be expanded on in terms of further experimental work? To start at the point this experiment ends at, the method used, specifically the two-iterations approach, can be changed trying to find higher improvement potential. For example, as hinted at in Section 5.6, the weighting between the stages might be suboptimal as the contribution in value changes are much higher in the first stage component. Moreover, more than two iterations could be run, for example in a way that every class is optimized for at least once. These iterations could potentially also be managed further with meta-heuristic methods. Another approach could be splitting the generation of the base solution in two steps, the first of which being calculating a good teacher-course assignment for a potentially massive decrease in runtime in the second split that generates the base solution.

6.3 Practical application

As we can see, there are lots of possibilities to improve the model as it is used in this experiment. A totally different way of improvement is trying to get the experiment closer to reality. This might not directly yield better results in terms of numbers, but it expands on the subject of how viable this topic really is in practice. The way this can be done is, on one hand, modelling the actual system of a school curriculum more precisely and using data from actual schools, or changing the method used to calculate optimal substitution plans. The probability distribution used to generate the scenarios could be replaced by one that is more backed by statistical data. Arguably, the assumption that every teacher is absent independently and with the same probability is not a very good reflection of reality. School policies might also be a factor to consider when designing such models, even though it might be hard to find a good generalization for these. Looking at these ways of improving, we can see that, right now, we are a step closer but yet still far from bringing the concept of timetables optimized for substitution plan quality to schools.

6.4 Personal lessons and experiences

Apart from all the future outlook, there are some aspects that can be learned from the experiment outside the results alone. Technically these are primarily things that I, as the author, have learned throughout preparing and producing this work. Most notably, the biggest trap to fall for in my opinion is thinking that the calculation will run within a realistic time. The approach of defining a model in a straightforward way and then

looking for ways to reduce calculation time by sacrificing precision is one that works, but it is probably worth to think about a model that already has some kind of trade-off for reducing calculation beforehand. In the end, from the model I originally planned to use not very much had made it into the final round.
Most other things I personally learned are more of the technical side. For example, I found out that generating two objects of the same type with the exact same values does not result in the two objects being considered equal in python by default. This results in quite some annoying interactions (that being no interaction), for example when applying a set of scenarios of which the teachers are not generated from the set of teachers in the timetable.
What this experiment also shows, especially in regards to what it tries to achieve, is that looking at a problem from a different angle can uncover new ways to improve the situation around it. To be specific in the context, the problem we looked at in the first place was the high rate of cancelled lessons in Bavarian schools. It is a topic that found continuous public attention for quite some time, alongside with the problem of too few teachers, issues with too big classes or other aspects about the quality of our educational systems. Reducing the number of cancelled lessons caused by teachers being ill did not seem to be an issue that had a high potential for improvement. And even though we technically did nothing to reduce this number itself, despite the fact this problem has been there for so long, we were still able to find a promising way to work on improving the situation around cancelled lessons by looking from another perspective - that is, by improving the quality of the substitution lessons, getting them closer to what the regular lessons would be.

7 Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorstehende schriftliche Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die benutzte Literatur sowie sonstige Hilfsquellen sind vollständig angegeben. Wörtlich oder dem Sinne nach dem Schrifttum oder dem Internet entnommene Stellen sind under Angabe der Quelle kenntlich gemacht.

8 Appendix A: Graphs

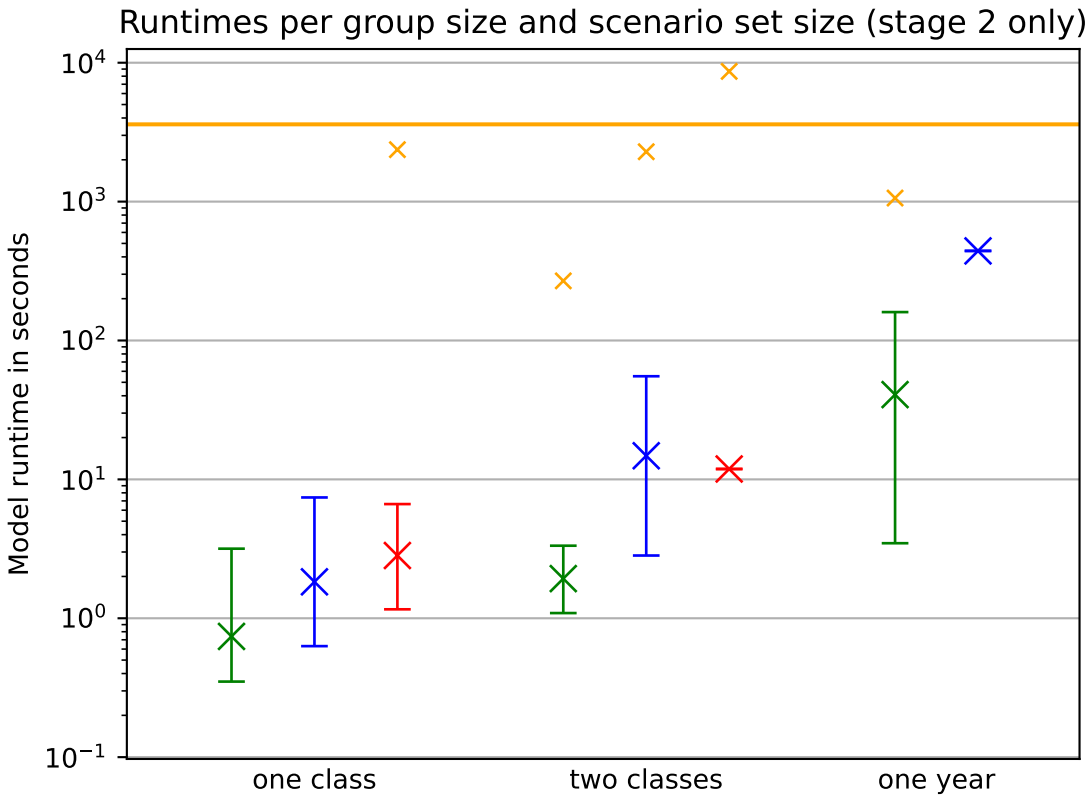


Fig. 8.1: The figure shows the runtimes of second iteration instances with the objective function optimizing for stage 2, showing the average, maximum and minimum outcomes. Green elements are instances with five scenarios, blue elements represent instances with 15 scenarios and red elements stand for 30 scenario instances. The horizontal line in orange is at the imposed time limit of 3600s. Additionally, orange points show the sum of average loading time and the average model runtime. For more precise numerical values as well as the number of instances run for each data point refer to Table 9.1 and Table 9.3

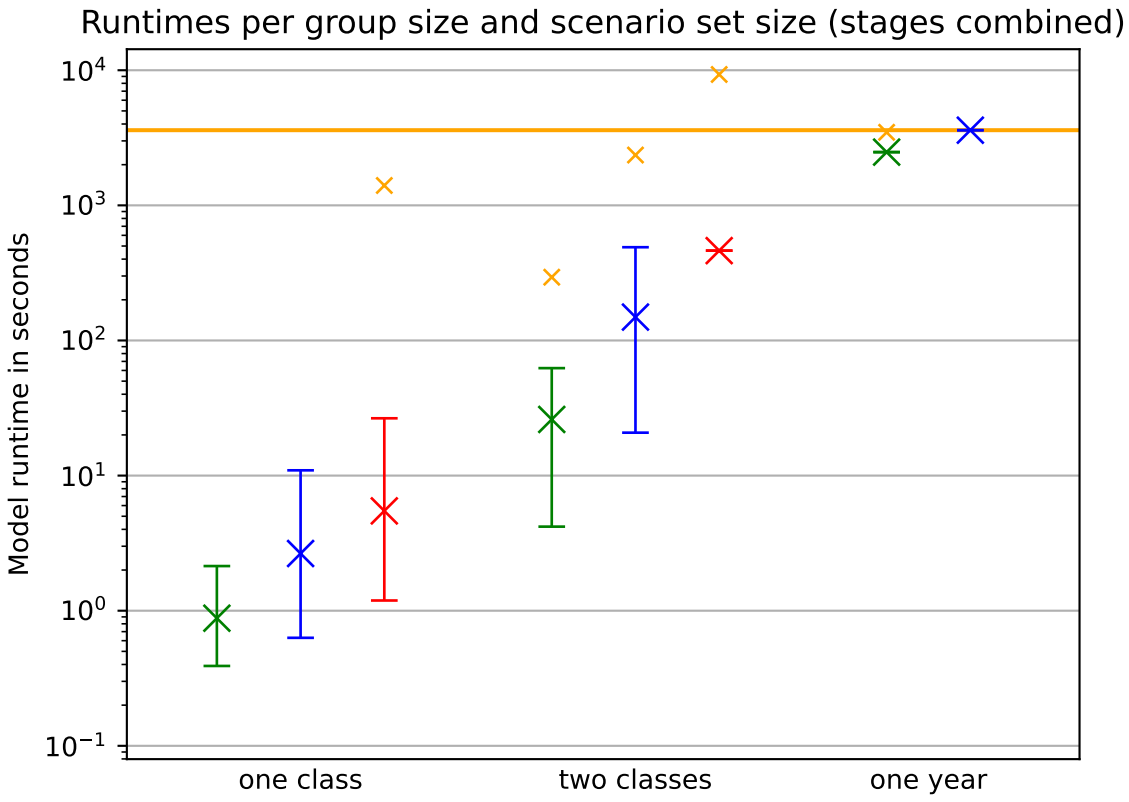


Fig. 8.2: The figure shows the runtimes of second iteration instances with the objective function optimizing for both stages combined, showing the average, maximum and minimum outcomes. Green elements are instances with five scenarios, blue elements represent instances with 15 scenarios and red elements stand for 30 scenario instances. The horizontal line in orange is at the imposed time limit of 3600s. Note that experiment with only one available data point have very high loading times and in theory hit the time limit before the model was ready to begin calculation. For more precise numerical values as well as the number of instances run for each data point refer to Table 9.1 and Table 9.4

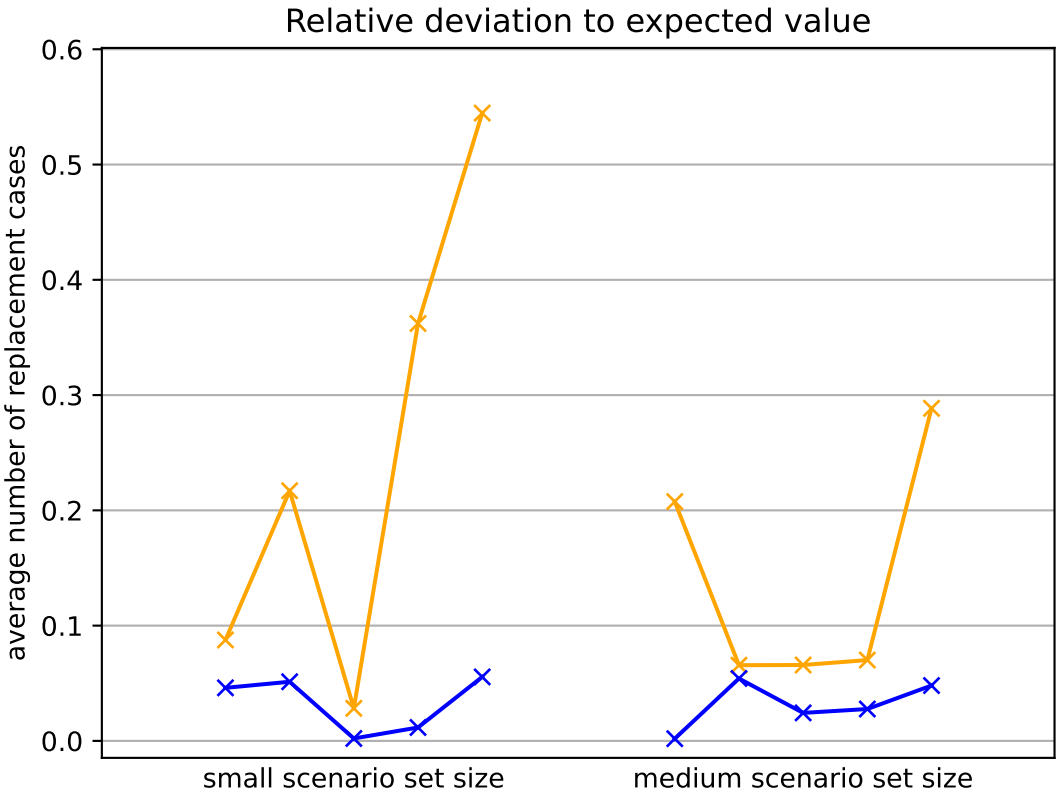


Fig. 8.3: The figure shows in orange the relative deviation per timetable of the reevaluation between the scenario set used for optimization and a separate random set of 100 scenarios. In blue there is also shown the relative deviation of a random subset of 50 of the large scenario set to the large scenario set.

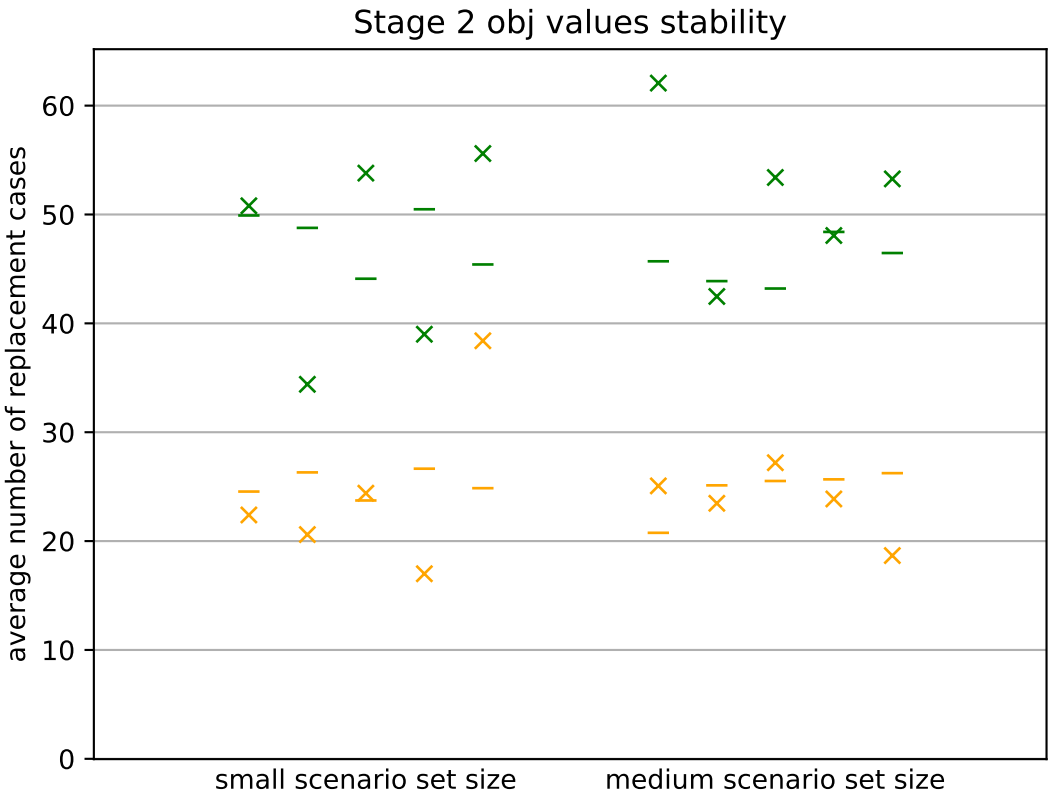


Fig. 8.4: The figure shows how often each replacement case (Section 1.2) happens at average with optimal substitution tables (according to Equation (4.32)) for several timetables as second iteration results. Green and orange x-markers represent the values obtained from reevaluating the timetable on the scenario set prior used for optimization, with green standing for same-subject replacements and orange for not-same-subject replacements. For the same colors, dash-markers show the values obtained from reevaluating the timetable on the 100-sized set, The case of no replacement has not occured in the scenarios and is therefore omitted. Note that this causes the value of the objective function to be proportional to the orange data (that is, if $pen^{V1} = 0$).

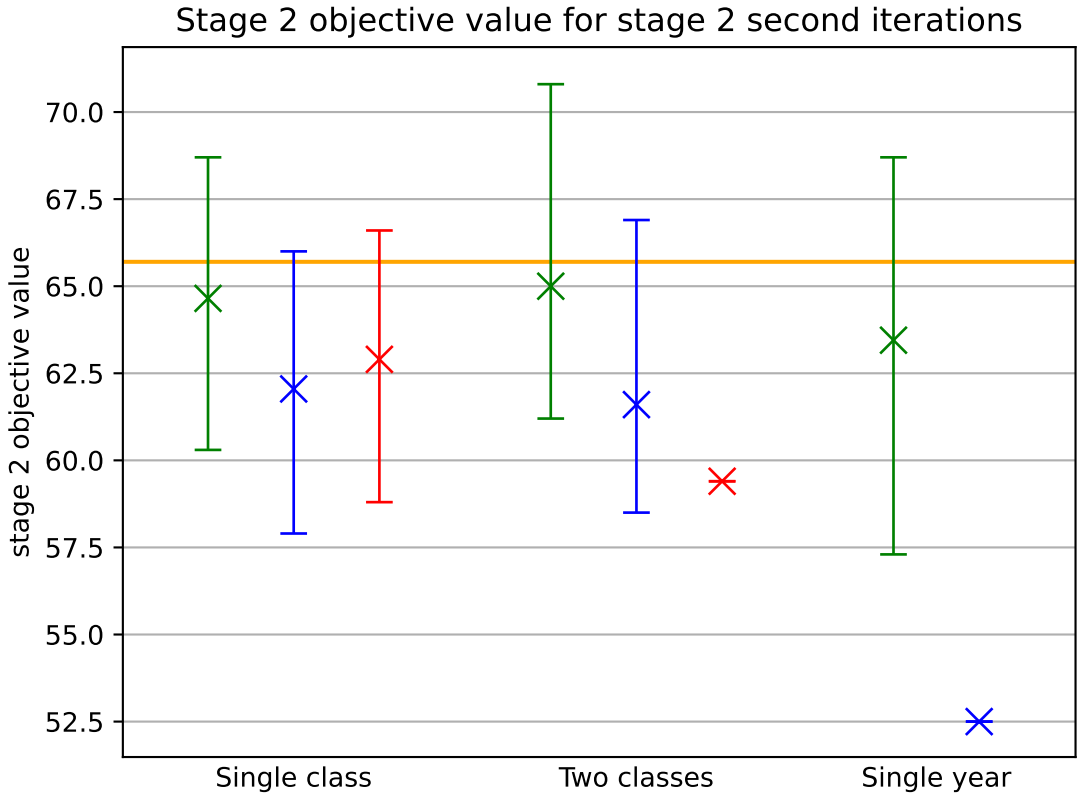


Fig. 8.5: The figure shows the stage 2 objective values of the second iterations that only optimized for stage 2. The colors represent the different sizes of scenario sets that have been used, with green being small (size 5), blue being medium (size 15), and red being large (size 30). The orange line shows the value of the base solution for comparison. The size of the dataset for each bar is 6. For the categories that show a single point, there is only one instance available due to high runtimes of instances of these categories (as shown in Figure 8.1)

9 Appendix B: Tables

Average runtime/gap for different instance categories

Instance description			avg runtime	avg gap	
obj. fn.	group size	|U|	(#solved)	(#unsolved)	solved %
stage 1 only	one class	–	0.45s (24)	0	100%
	two classes	–	11.14s (12)	0	100%
	one year	–	1487.51s (4)	0.035 (2)	67%
stage 2 only	one class	5	0.74s (24)	0	100%
		15	1.83s (24)	0	100%
		30	2.83s (6)	0	100%
	two classes	5	1.93s (12)	0	100%
		15	14.80s (6)	0	100%
		30	11.88s (1)	0	100%
	one year	5	40.78s (6)	0	100%
		15	441.64s (1)	0	100%
stages combined	one class	5	0.89s (24)	0	100%
		15	2.65s (24)	0	100%
		30	5.48s (6)	0	100%
	two classes	5	26.03s (12)	0	100%
		15	148.97s (6)	0	100%
		30	650.50s (1)	0	100%
	one year	5	2477.33s (1)	0.049 (5)	17%
		15	0	0.085 (1)	0%

Tab. 9.1: This table shows the runtimes and gap averages for all categories. A zero in a cell of the column "avg runtime" means there was no instance of the experiment category to successfully solve within the time limit. A zero in a cell of the column "avg gap" means that all instances of the experiment category successfully solved within the time limit.

Avg/min/max runtime for stage 1 runs

group size	average runtime	minimal runtime	maximal runtime	# solved instances
one class	0.45s	0.27s	0.99s	24
two classes	11.14s	1.27s	48.58s	12
one year	1487.51s	32.99s	3600.00s	4

Tab. 9.2: This table shows the average, minimal and maximal runtime values of the stage 1 second iteration runs. Note that two instances of the group size "one year" have not been solved successfully within the time limit and are thus not included in the average of the category

Avg/min/max runtime and avg loading time for stage 2 runs

group size	|U|	avg runtime	min runtime	max runtime	# solved instances	avg loading time
one class	5	0.74s	0.36s	3.17s	24	not measured
	15	1.83s	0.63s	7.41s	24	not measured
	30	2.83s	1.16s	6.64s	6	2363.98s
two classes	5	1.93s	1.09s	3.33s	12	260.77s
	15	14.80s	2.83s	55.28s	6	2271.46s
	30	11.88s	11.88s	11.88s	1	8650.51s
one year	5	40.78s	3.47s	160.13	6	1020.12s
	15	441.64s	441.64s	441.64s	1	not measured

Tab. 9.3: This table shows the average, minimal and maximal runtime values of the stage 2 second iteration runs, along with the number of solved instances and the average loading time

Avg/min/max runtime and avg loading time for combined stages runs

group size	|U|	avg runtime	min runtime	max runtime	# solved instances	avg loading time
one class	5	0.89s	0.39s	2.14s	24	not measured
	15	2.65s	0.63s	10.94s	24	not measured
	30	5.48s	1.19s	26.55s	6	1379.50s
two classes	5	26.03s	4.19s	62.41s	12	267.92s
	15	148.97s	20.78s	489.91s	6	2208.29s
	30	462.36s	462.36s	462.36s	1	8841.90s
one year	5	2477.33s	2477.33s	2477.33s	1 (of 6)	994.95s

Tab. 9.4: This table shows the average, minimal and maximal runtime values of the stages combined second iteration runs, along with the number of solved instances and the average loading time. Note that in the row for group size one year and $|U| = 5$ the average runtime is calculated from the solved solutions only, whereas the average loading time is calculated from all runs made. The average gap of the unsuccessful runs of this category is about 0.49

Avg stage 1 objective value improvements (with components) for stage 1 runs

		weekly distribution	double lessons	priority courses	total stage 1 objective value
base solution		438	-90	-386	-38
one class	avg. value	427.0	-104.2	-395.3	-72.5
	abs. improvement	-11.0	-14.2	-9.3	-34.5
	rel. improvement	-2.5%	+15.7%	+2.4%	-
	maximum/minimum objective value:				-66/-81
two classes	avg. value	419.5	-118.3	-405.0	-103.8
	abs. improvement	-18.5	-28.3	-19.0	-65.8
	rel. improvement	-4.2%	+31.4%	+4.9%	-
	maximum/minimum objective value:				-97/-112
one year	avg. value	396.5	-170.8	-432.7	-207.0
	abs. improvement	-41.5	-80.8	-46,7	-169,7
	rel. improvement	-9.5%	+89.8%	+12.1%	-
	maximum/minimum objective value:				-150/-238

Tab. 9.5: This table shows a comparison of the stage 1 components of the objective function between the base solution and the averages of six second iterations for each group size as well as absolute and relative improvement. Note that the objective function is a minimization function, so negative absolute improvement is good, and positive absolute values get better with negative relative improvement whereas negative absolute values do so with positive relative improvement. As the objective value for stage 1 is a combination of positive and negative components, a relative improvement measurement is not very informative

Avg objective values by stages for combined stage runs

Instance description			avg first stage	avg second stage	avg total
obj. fn.	group size	|U|	obj. value	obj. value	obj. value
base solution			-38	65.7	27.7
stage 1 only	one class	–	-72.5	65.0	-7.5
	two classes	–	-103.8	65.8	-38.0
	one year	–	-207.0	67.9	-139.1
stage 2 only	one class	5	-44.3	64.7	20.4
		15	-42.5	62.1	19.6
		30	-41.7	62.9	21.2
	two classes	5	-52.5	65.0	12.5
		15	-51.8	61.6	10.2
		30	-52.0	59.4	7.4
	one year	5	-60.5	63.5	3.0
		15	-37	52.5	15.5
stages combined	one class	5	-72.2	64.7	-7.5
		15	-72.5	63.5	-9.0
		30	-69.3	61.6	-7.7
	two classes	5	-103.8	63.7	-40.1
		15	-103.8	61.9	-41.9
		30	-104.0	61.2	-42.8
	one year	5	-205.3	63.6	-141.7
		15	-148	56.4	-91.6

Tab. 9.6: This table shows the average objective values for first, second and combined stages for each group of instances. Note that some rows, namely the stage-1/one-year and stages-combined/one year data are subject to results of models not solved to optimality within the time limit of 3600s. Moreover, the rows stage-2/one-year/15 and stages-combined/one-year/15 only have one instance for data, as opposed to all other rows having six.

Bibliography

[BCRT15] Andrea Bettinelli, Valentina Cacchiani, Roberto Roberti, and Paolo Toth: An overview of curriculum-based course timetabling. *Top*, 23:313–349, 2015, 10.1007/s11750-015-0366-z. https://doi.org/10.1007/s11750-015-0366-z.

[BL11] John R. Birge and François Louveaux: *Introduction to stochastic programming*. Springer New York, NY, 2nd edition, 2011, 10.1007/978-1-4614-0237-4. https://doi.org/10.1007/978-1-4614-0237-4.

[BMPR10] Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová: Decomposition, reformulation, and diving in university course timetabling. *Computers &amp Operations Research*, 37(3):582–597, mar 2010, 10.1016/j.cor.2009.02.023. https://doi.org/10.1016%2Fj.cor.2009.02.023.

[Boh23] Katrin Bohlmann: "es reicht!": Warum immer weniger menschen lehrer werden wollen, 2023. https://www.tagesschau.de/inland/regional/bayern/br-es-reicht-warum-immer-weniger-menschen-lehrer-werden-wollen-102.html, Last accessed 21 August 2023.

[DGMS07] Luca Di Gaspero, Barry Mccollum, and Andrea Schaerf: The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). January 2007.

[FW22] Leiter Öffentlichkeitsarbeit des Bayerischen Staatsministeriums für Unterricht und Kultus Fabian Welling: Übersicht zum unterrichtsausfall (schuljahr 2021/2022), 2022. https://www.km.bayern.de/lehrer/unterrichtsversorgung/zahlen/lehrpersonalbedarf.html, Last accessed 16 August 2023.

[GH09] A. A. Ahmadi G. Hall: Robust optimization, year unknown (>2009). http://www.princeton.edu/~aaa/Public/Teaching/ORF523/ORF523_Lec16.pdf, Last accessed 25 August 2023.

[Gri22] Markus Grill: Schulschliessungen in der pandemie - eine chronologie, 2022. https://www.markusgrill.eu/2022/02/10/schulschliessungen-in-der-pandemie-eine-rekonstruktion/, Last accessed 21 August 2023.

[Roc01] R. T. Rockafellar: Optimization under uncertainty, 2001. https://sites.math.washington.edu/~rtr/uncertainty.pdf, Last accessed 25 August 2023.

[Son13] Heike Sonnberger: Es fallen dauernd stunden aus - oder doch nicht?, 2013. https://www.spiegel.de/lebenundlernen/schule/stundenzahl-und-unterrichtsausfall-so-viel-lernen-schueler-a-877501.html, Last accessed 21 August 2023.

[Spo15] Peter Sport: Lasst sie doch noch etwas schlafen, 2015. https://www.faz.net/aktuell/feuilleton/familie/schulunterricht-bei-jugendlichen-soll-spaeter-beginnen-13436308.html?printPagedArticle=true#pageIndex_3, Last accessed 23 August 2023.

[SS80] G. Schmidt and T. Stroehlein: Timetable construction - an annotated bibliography. *The Computer Journal*, 23(4):307–316, January 1980, 10.1093/comjnl/23.4.307, ISSN 0010-4620. https://doi.org/10.1093/comjnl/23.4.307.