Timetabling for on demand learning: models and algorithms

Timetabling for on demand learning: models and algorithms

Jos Roseboom

Contents

Preface

This thesis is the result of my final project for the master program "Applied Computing Science" of the department of "Information and Computing Sciences" of Utrecht University (Utrecht). For this final project, I worked with Educator BV for nine months.

I wish to thank all the people who's help contributed to this thesis. I thank all my Educator colleagues for making my work joyful and providing me with ideas. I thank Vincent Kok, who was my supervisor at the company, for his useful feedback. I thank Marjan van den Akker for being my supervisor of Utrecht University. Her insights were crucial. I thank my girlfriend Saskia for her support during the project.

From kindergarten to university, a lot of people supported me some of the time. Two supported me all the time. Special thanks go out to my parents, Kees and Joke.

Chapter 1

Introduction

1.1 Problem background

The way education is organized changed rapidly in the past years, and is still subject to changes. Electronic support has grown in popularity, now that having a computer with internet connection is common. This development goes hand in hand with a change in education design. Traditionally, education was based on a model of classes, with those classes walking a predefined route, such that the members of the class master a profession at the end of this route. Nowadays, education is focused more and more on the students individually. Every student should be able to define his own route, selecting his courses from the course catalog. The school has to make sure there is enough capacity for the demand. This concept is referred to as on demand learning. Educator facilitates this process for example by a very flexible course catalog, possibilities for a mentor to monitor the progress of a student and, for each student, a personal activity plan. A gap in the Educator application is the absence of a timetabling engine that is able to generate timetables respecting the urges and needs of on demand learning.

1.2 Company background

Educator BV is a company providing a software solution for managing human development. By using the software package of Educator BV, "Educator", organizations get clear insight in the development of its people. Often, those are organizations that have this human development as their primary concern, like schools. Although the ambition of Educator BV is to serve a broader field of organizations, current costumers of Educator BV are all schools. Some costumers of Educator BV are:

- Windesheim (Zwolle)
- Drenthe College (Drenthe)

- NHL (Leeuwarden)

Educator BV operates under the wings of Cordys BV, which is part of the "Vanenburg Investment Group", which is owned by the "Oikonomos" welfare foundation. More information about Educator can be found on the web: http://www.educator.eu.

1.3 Current situation

At the moment, Educator does not have any timetable engine. Costumers of Educator use their own engine next to Educator. A major costumer of Educator, Windesheim, uses Untis now, after using Iris for a while. Educator has tried to cooperate with Iris, but this has failed. The requirements for a timetable engine are not clear to Educator. Clearing this out is part of the project.

1.4 Project definition

One of the purposes of this project is to provide Educator with a prototype for a timetabling engine. This prototype uses a local search technique. Educator also wants to be provided with information on general components in a timetable process. It must be clear what to optimize. Within the project, the following questions need to be answered (explicitly requested by Educator BV):

- Which optimization goals are important for a school?
- What is the relation between the scheduling of teachers and timetable optimization?
- What are the resources of relevance for a school?
- What interaction is needed during optimization?
- How to quantify a timetable, and make corresponding choices?
- Provide Educator with knowledge that can be used as a base to decide whether or not to build its own timetable engine.
- Build a prototype and validate this prototype with some of Educator its customers.

Chapter 2

Problem exploration

This chapter gives an overview of the process of defining the problem, and defining a good scenario to solve it. First of all, the initial vision of Education is described. Using this vision as a starting point, some scenarios were composed. These scenarios were presented and adapted a few times, which is summarized in this chapter, and described in full detail in the Appendix. The final scenario is presented in full detail in this chapter. This scenario, approved by both Educator and Utrecht University, is the base of the project.

2.1 Course and lesson

Entities like teacher, room, student, etc. have a clear meaning. However, the distinction between a course and a lesson is not that straightforward. For a proper understanding of the remainder of this chapter, this distinction needs to be clear.

2.1.1 Course

A course is an entity a student can subscribe to. Completing a course successfully will most often provide this student with some credit points (e.g. ECTS), of which a certain amount is needed to complete his program. A course exists of lessons and exams. Exams, however, are not considered here. The lessons can be of various type (lecture, practice, etc.). We will refer to this as the lessontype.

2.1.2 Lesson

A lesson is an entity belonging to a course. A student should participate in such a lesson a certain number of hours a week. This number is defined in the definition of the course this lesson belongs to, e.g. "lecture" should be provided two hours in a week. To provide enough capacity for a lesson, multiple instances of this lesson may be offered during the week.

2.2 Vision Educator

Educator has tried to tackle the problem. A schematic view of Educator its idea is given in figure 2.1. Education design is the structure of the provided education. This

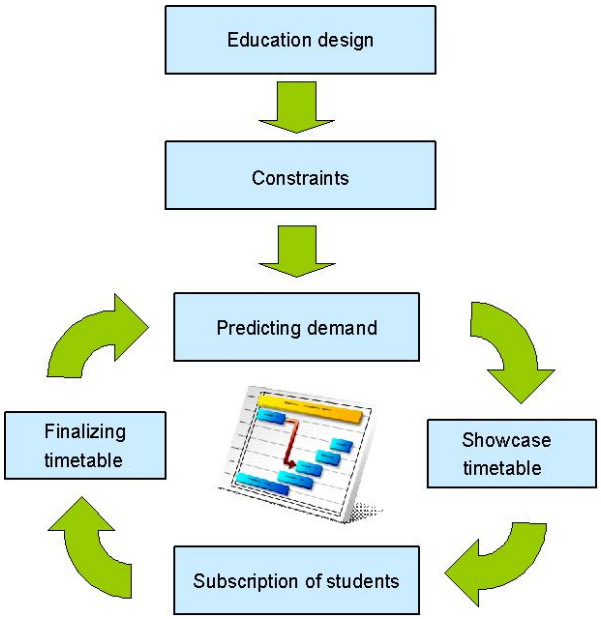


Figure 2.1: Schematic view vision Educator

is given by the taught courses and their coherence (a course is for example part of a minor). In the traditional education design, every student has a learning route. A learning route is a package of related courses, to master a profession. Within the current development in education arises the desire to drop the concept of a required learning route. Students still can have a required learning route, but on demand learning must also be possible. In the concept of on demand learning, a student does not have a predefined route of courses to participate in, but decides every subscription what courses to participate in. The curriculum of offered courses is given, and from that point, on demand learning is provided. In figure 2.1, the constraints are the requirements in a timetable (for example, the availability of teachers). The prediction of demand for courses is based on the attendance in courses in the past years. Based on this prediction, a showcase timetable is generated. A showcase timetable is an almost final timetable, published prior to subscription. In the vision of Educator, this showcase timetable is very precise. This showcase timetable has assigned a time, a teacher and a room to the lessons of the scheduled courses. The only changes that can be made after subscription of students, is the addition of lessons (finalizing the timetable) and removal of lessons without subscripted students. Times, teachers and rooms assigned in the showcase timetable need to be respected for lessons for which at least one student has been subscribed (even if this is only one student).

2.3 Development

To come to a mature scenario, we composed a number of different possible ways to tackle the problem. As a start, three scenarios were considered. These scenarios were discussed at Utrecht University and Educator BV. The scenarios and their discussion are described in full detail in Appendix A.

After adapting the selected scenario (based on the discussion), the resulting provisional scenario was presented to a domain expert of a department ("School of Health Care") of Windesheim, a major customer of Educator BV. An objective of this visit was to get some reflection on the scenario. Another objective was to investigate the current problems with timetabling as well as the problems of Educator to cooperate with the timetabling software of Windesheim. A detailed report of this visit can be found at Appendix B.1.

To get a broader perspective, a second visit was paid to Windesheim. This time, a member of the central management of Windesheim was interviewed. Full details of this visit can be found in Appendix B.2.

2.4 Final Scenario

2.4.1 Time

Time units considered generating timetables are:

- Planning cycle (e.g. a year): One round of a cycle of more or less the same curricula. This cycle contains one or more education periods.
- Education period(e.g. a quarter of a year): The period a course takes.
- Lesson cycle (e.g. a week): Within an education period, often a same schedule is repeated. The time this takes is a lesson cycle.
- Lesson option (e.g. an hour): a time unit at which a lesson can take place.

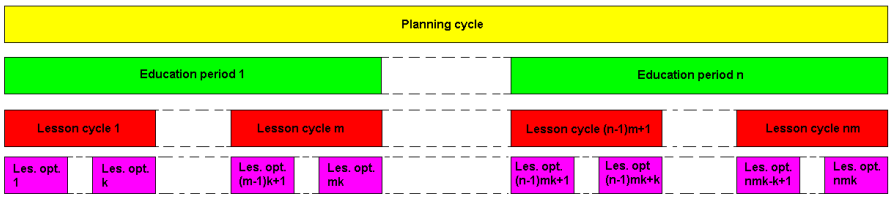


Figure 2.2: Time

2.4.2 Process

We will point out the process of the final scenario using some diagrams. The first diagram shows the entire process. Later diagrams will go more into the details of a part of the process that was presented as a black box in the previous diagram.

Complete process

Figure 2.3 shows the complete process. At first, a distribution of courses among the education periods is made. Data that has to be available as input for this distribution is:

- The courses that should be taught this plancycle.
- Some general information, like the relation of the time units described in section 2.4.1.
- The available resources to facilitate education.
- Some statistical information to predict the demand for the courses.
- The students that possibly participate in the courses.

As a result, every education period has a number of suggested courses attached to it. This suggestion is forwarded to the education period scheduling process. To decide which distribution is the best, resources are coupled to lessons. This is only done to check the feasibility and determine the quality (since resource utilization factor is a measure for quality), so this underlying attachment of resources is not given in the output (hence the minus sign in the figure 2.3). The quality of a distribution determined by:

- A student is able to fill the planning cycle with a maximum amount of courses fitting his profile (not a half occupied education period preceded by an education period with much overlap)
- Resources are equally used during the planning cycle

In "education period scheduling" process, the scheduling master can decide which courses he wants to schedule that period. He is recommended not to deviate from the suggestion.

Education period scheduling

In figure 2.3, the scheduling of an education period is a black box. Figure 2.4 reveals some details of the process. The input data is the same as in figure 2.3 together with the suggestion that resulted from the plancycle distribution. This input is used to generate a timetable for the first week. In current practice, every week timetable is equal within the same education period. However, this is not mandatory. After a week, the scheduling master can decide to copy the timetable to the next week, or to adapt this timetable. The current timetable will be used as a base timetable in case of adaptation. The scheduling master enters some timetable requirements and a timetable for the next week is generated.

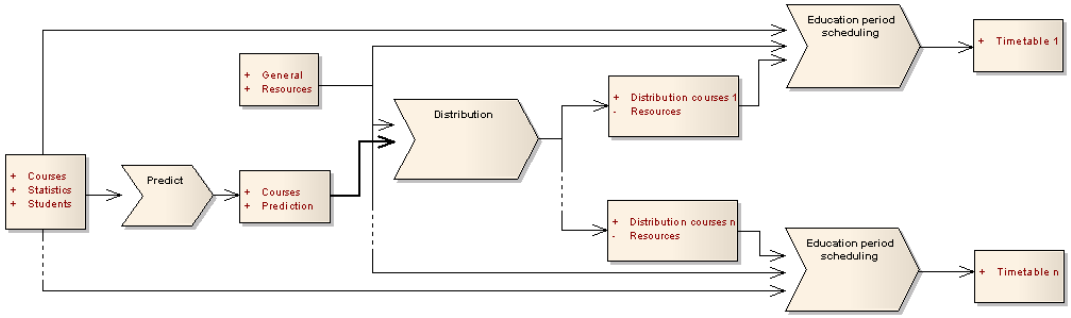


Figure 2.3: Overview of the complete process

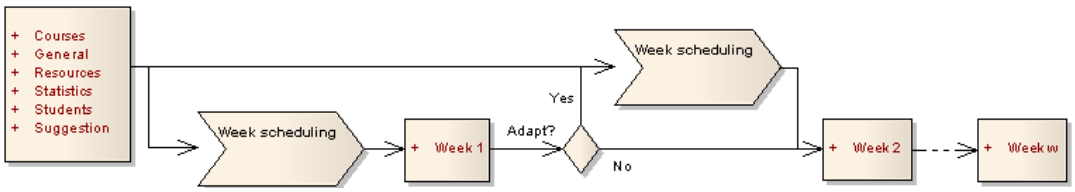


Figure 2.4: Overview of education period scheduling

Weekscheduling

In figure 2.4 the scheduling of a week is a black box. Here, we go into more detail, as shown in figure 2.5. At first, the schedule master has to decide which courses there are to be scheduled. Input for making this decision are the courses possible to schedule, and the suggestion. Furthermore, the schedule master enters some configuration parameters used during the process. The weekschedule can be based on an existing schedule, that is loaded from memory ("Base schedule" input parameter) or the result from the previous iteration (of which the scheduling master is not fully satisfied). After this, the courses that are to be scheduled are known. Together with the students that possibly participate in education this week and some statistical information, a prediction is made for the consumption of courses this week. The prediction is suggested to the scheduling master who confirms this prediction, or adapts it. Together with the configuration entered and the available resources, a week timetable is generated by the timetable engine. Final result for this iteration is a week timetable with some information indicating the quality, and a timetable for each student that is the best possible timetable if the objective is to give all the students a timetable of the best possible quality. This result is considered by the schedule master. He can decide to accept this solution or reject this solution.
In the first case, the current solution will be saved and the result will be published. If later on, the timetable is not appropriate anymore (for example due to a contrast in real participation and expectation), adaptations can be made by calling the weekscheduling routine again. The current solution will be loaded from memory and used as a base schedule.
In the latter case, the weekscheduling routine is called again, using the current solution as a base schedule.

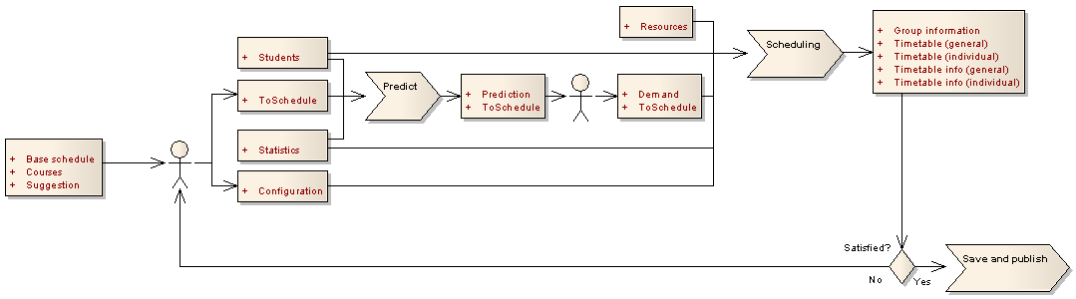


Figure 2.5: Scheduling of a week

Predict

The process of predicting is a black box in figure 2.5. More details on this can be found in figure 2.6. Based on the possible students, the courses that are offered and statistical information, a number of simulations are done. Each of those simulations has a set of students as a result. This set contains all the possible students (given as input) with a set of courses attached to them. This set is a set of courses this student is expected to participate in. A prediction is computed to count for every combination of courses the number of students that participate in both of them. Those numbers are divided by the number of simulations ran. Hence, the result is a table of $c \times c$, where c is the number of courses. The cells express the expected number of students that participate in both courses. On the main diagonal are the expected students per course (since this expresses the overlap of a course with itself).

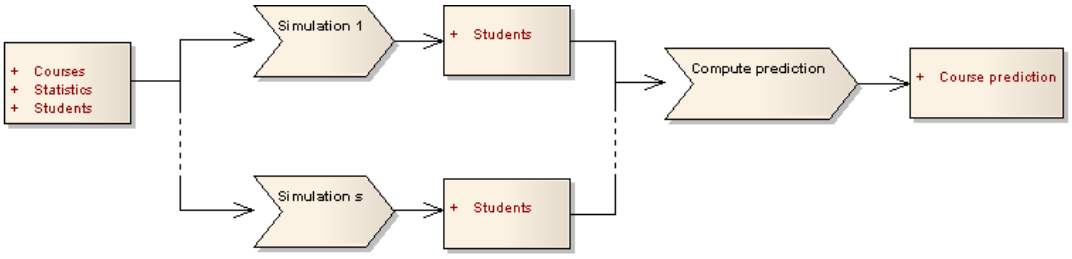


Figure 2.6: Predict demand

Scheduling

The scheduling process itself is a black box in figure 2.5. Details are given in figure 2.7. A timetable is generated using the courses to schedule with their corresponding demands, the configuration that was entered by the schedule master and the available resources. Result is a plain timetable; just times, lesson and resources, no additional information. The quality of this timetable is determined by:

- Idle time of students as little as possible
- Not to little idle time for teachers
- Using resources as little as possible above a certain utilization factor

- Acknowledging the preferences concerning time and courses of the students
- Travel time between lessons that are not separated by a break as little as possible
- Acknowledging the lesson preferences of the teachers

A number of new simulations are done, each resulting in a set of students with each student having a set of courses this student is expected to participate in. Each of those sets are matched to a different copy of the plain timetable. The quality of a matching is determined by:

- Idle time of matched students as little as possible
- Students have their lessons at preferred moments
- The length of a day of education is conform the students preferences
- Traveltime above a threshold as little as possible
- It is possible to form groups such that students can participate in a set of lessons with that group

This matching process results in a timetable for each student with corresponding information about this timetable (idle time etc.), some group information and some general timetable information. Group information lists the set of students per combination of lessons that participate in all lessons of the combination. The combinations, as well as the requirements to the size of these combinations, are part of the entered configuration information. For example, the lesson "project" has to be taken two hours a week, and in groups of four to six students. Now suppose "project" is offered ten times a week, each combination of instances of the lesson "project" needs to be of a size such that groups of size four to six can be composed. The general timetable information contains information like the total idle time of all the students. This general information is averaged over all the student matching results, and used as timetable information.

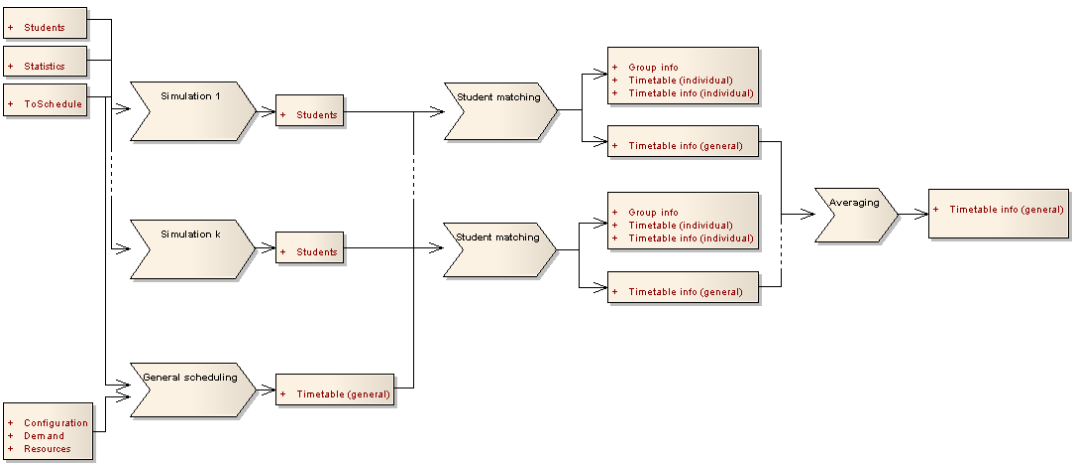


Figure 2.7: Actual scheduling process

2.4.3 Exams

A week containing only exams will be scheduled separately. If an exam is to be taken within the education period, and another room than the usual one is required for this, this room can be requested for the week the exam should take place. If one of the rooms used for one of the lessons of this course suffices, the fact that it concerns an exam can be ignored while generating the timetable.

2.4.4 Windesheim version

This scenario could be applied to the Windesheim situation. The planning cycle at Windesheim is not the same for every school. For example, at the School of Health Care, the planning cycle is a semester (half a year), while at the School of Information Sciences this is a year. The students subscribe for the entire semester at the Windesheim schools. Exams take place in the last two weeks of an education period and have their own timetable. The weeks in the education period prior to the exams do not contain exams.

Chapter 3

Literature overview

This chapter gives a brief overview of some literature related to the scope of this thesis. Each paragraph gives a slight introduction to concerned paper.

In physics, in the field of statistical mechanics, a technique is used to simulate the behavior of a group of particles under different temperatures. Within this simulation, the behavior of the group is simulated without considering the behavior of the individual particles. Kirckpatrick et al [KGV83] present a parallel to the field of combinatorial optimization.

In [ECF97], a comparison is made of different annealing techniques used to solve a timetabling problem much resembling the one of this thesis. The problem is stated briefly and objectives are not quantified. Emphasized in this paper is the optimization technique. Simulated annealing turned out to provide the best results. Withing simulated annealing, experiments have been done to find out the best cooling technique.

In [MW66] is discussed how to organize the subscription procedure given an existing timetable. Two strategies are discussed: first come first serve and assigning to the course occurrence with lowest utilization factor (subscription is on course, not a specific course occurrence). Implementation of the first strategy is trivial, implementation of the second is further discussed.

Hyper-heuristics are heuristics that are used to decide what heuristic to use. In [BBKM06], this idea is discussed, in combination with the optimization technique of simulated annealing. Here, the low level heuristic is the heuristic that transforms the current solution to a neighbor in the solution space. There are multiple heuristics that could be applied for that purpose. To decide which to use, a heuristic is used (the hyper heuristic). Basically, this hyper heuristic adapts the probability of a low level heuristic being applied. This is done corresponding to the result of the application of this low level heuristic.

A meta-heuristic is an heuristic that solves a general problem. The paper [Lew07] discusses approaches for the general planning problem. This could be university timetabling, but nurse scheduling in hospitals as well. Such a meta heuristic can be for example simulated annealing or tabu search. Three meta heuristic categories are discussed: one-stage optimization algorithms, two-stage optimization algorithms, algorithms that allow relaxations. In one-stage optimization algorithms, hard constraints and soft constraints are being satisfied simultaneously. Two-stage optimization algorithms first satisfy the hard constraints. Once they are satisfied, care is taken for the soft constraints. Algorithms that allow relaxations allow postponed scheduling if not all events can be scheduled. Soft constraints are attempted to satisfy, and at a later stage, the postponed events are attempted to schedule. The paper gives an overview on the techniques and gives a broad range of references.

The paper of De Werra [dW85] states that there are in general two distinct phases in timetabling. At first, the curricula are defined for each group of students (class) and resources are assigned to the classes. Secondly, a workable timetable for the class-resources combinations is searched for. In the paper, first a way is described on how to couple classes to teachers. After this, the problem is introduced of minimizing the number of days to schedule all the couples. Last problem introduced here accepts the days as they are given, and discusses how classes and teacher can be spread as much as possible among the available days. Methods to solve those problems are explained, most often by reducing it to a coloring or flow problem.

Chapter 4

Data

Generating a timetable requires data. This chapter discusses the data that is required and the relation of the different data elements; the data model. To test the application, a testworld is generated, which is described in the second section of this chapter.

4.1 The data model

The data elements and their relations are shown in Appendix C. For readability of this section, it is recommended to keep an eye on Appendix C. In this section, the elements of the diagram are discussed in detail.

4.1.1 Availability

Availability data stores availability for one week. Availability has a field expressing the week it stores the availability for, and a field expressing the usable hours. The number of usable hours can differ from the number of available hours. For example, a teacher can be available 40 hours, while he is only willing to teach 32 hours. Keeping track of these fields separately instead of forcing the teacher to select 32 out of the 40 hours (while he might be indifferent) allows more possibilities, and thereby increases the possibility for a satisfying timetable. Availability has a WeekTimeUnit for every time unit of the week. This unit expresses whether or not this time unit of the week is available, and how popular the use of this time unit is. For example, this preferability field can be used to express a preference for a lesson of a certain type to take place after noon.

4.1.2 General data

General data stores the duration of the time periods used for scheduling, as defined in section 2.4.1. The fields in general data store the duration in weeks of the time unit that intuitively corresponds to its name. General data has also a relation to "Availability", that expresses the general availability of education. In other words, this expresses the allowed timespan for education to be scheduled.

4.1.3 Resource

Resource data stores the properties that all resources have in common. A resource is defined as anything that can be necessary to facilitate a lesson. Those commonly properties are: having an availability, an id and a perfect utilization. A perfect utilization for a resource is the highest utilization factor for which it is no longer considered an improvement to decrease the use of that resource. Utilization factor is defined as the ratio of the current use of a resource and the possible use of the resource. For example, when a teacher can be used 32 hours, and this teacher is currently used 24 hours, the corresponding utilization factor is $\frac{24}{32} = 0.75 = 75\%$. This perfect utilization can change over time. A realistic scenario seems that the planning master, in a premature stage of timetabling, considers it usefull to have some backup capacity for certain resources. For example, three weeks prior to the start of the resulting timetable, the first lessons are scheduled. The planning master expects the urge of adding lessons in a later stage, but currently the expected demand for these lessons is very uncertain. At this stage, the planning master could decide to add a preferability of using only 70% of the resource. Later, when more information has caused a sharper view of the expected demand, the planning master increases this parameter of the particular resource. Every resource has an availability, expressing when this resource is available, and how many time units the resource can be used. In the data model, the example resources "room" and "teacher" are considered, but this can be anything.

4.1.4 Course

Course data stores the data for courses. Fields of this element are the id of this course, the minimum and maximum number of education periods this course should be offered in, and the current expectation of the amount of students that will participate in this course. A course data element has two relations to other course data elements. The relation "must succeed" expresses possible precedence relations,i.e., when a course cannot take place if some other course has not taken place yet in the current planning cycle. Similar to this relation, "cannot be together" means the courses cannot take place in the same education period. A course data element consists of lessons that are to be taught. Section 4.1.5 will go into this in more detail.

4.1.5 LessonDefinition

A course has a few types of lessons that are to be taught. A type is typically something like "'lecture"', "'practical session"', etcetera. The properties of every type are stored in LessonDefinition . The fields of this data element are:

classHours	The hours of class that is required for this type of lesson.
expTypeDemand	The currently expected demand for this type of lesson.
id	The id of this type.
maxClassDuration	The maximum number of hours a lesson can take (upper-bounded by classHours).
maxStudents	The maximum number of students for a lesson of this type.
minClassDuration	The minimum hours a class should take (lower-bounded by 1). Used for multiple hour lessons.
minStudents	The minimum number of students for a lesson of this type.
perfectUtilization	This field is similar to the perfectUtilization field in section 4.1.3. In this situation, the concerned utilization factor is the ratio of used student capacity and offered student capacity of this type of lesson. The perfectUtilization field expresses the utilization factor of the situation in which more overhead in student capacity is not considered an improvement.

LessonDefinition has a relation to Availability, with a trivial meaning (if section 4.1.1 is known). The relations to itself are interpreted similar as "'must succeed"' and "'cannot be together"' of the Course data element. Difference is that those relations at the course data element concern education periods as time unit, while the relations of LessonDefinition concern the WeekTimeUnit, e.g., an hour, as a time unit. Offering one hour of a lesson of this type requires resources facilitating it, expressed in the "'requires"' relation to Resource. More often than not, multiple kinds of resources are required, with each kind having multiple candidate resources available to fulfill that requirement. However, not all the resources of that kind are needed to fulfill the requirement. Hence, the relation is a two dimensional one:

$$\text{Requirements} \;:\; \overbrace{\text{ResourceType}_1^S* \;\wedge\; \ldots \;\wedge\; \text{ResourceType}_n^S*}^{r_a, r_j, r_z}$$

The subscript number of ResourceType indicates the type of resource and the asterisk means there are 0 or more resources of that type required. The superscript 'S' means suitable for the considered lesson type. For example, suppose ResourceType_1 in the above represents teachers, and its members are:

$$\overbrace{\text{ResourceType}_1}^{\text{teachers}} \;:\; \{Ludwig, Wolfgang, Sergej, Albert, Isaac\}$$

Further, suppose there is a course "'opera"' and the type of the lesson we consider is "'lecture"', which has a LessonDefinition. For a "'lecture"' of "'opera"', one resource of type "'teacher"' is required. The teachers not qualified for this are Albert and Isaac, which makes the set of suitable resources of type "'teachers"':

$$\overbrace{\text{ResourceType}_1^S}^{\text{qualified teachers}} \;:\; \{Ludwig, Wolfgang, Sergej\}$$

4.1.6 ScheduledLesson

A LessonDefinition has a relation to ScheduledLesson. A ScheduledLesson is a lesson as it occurs in the timetable. A ScheduledLesson has a relation to the resources that facilitate the lesson. These resources are a subset of the required resources. In the example considered above, a ScheduledLesson belonging to a LessonDefinition of "'lecture"' of course "'opera"' could be facilitated by resource "'Ludwig"'.

4.1.7 Student

The only relevant field for a student is an id. A student data element has two relations to the Course data element. Trivially, "'has past"', expresses the courses that are past by this students. The other relation, "'intends to follow"', expresses which courses the students explicitly showed his interest for (e.g. by means of a survey). Significance of these relations concern demand prediction. If a student explicitly showed his interest for a course, or if a student has already past a course, this will clearly have its consequences for the expected demand for that course. The relation to PersonalTimeTable expresses the timetable for this student. A PersonalTimeTable data element consists of lessons with a moment in the week attached to it. Only the lessons for the concerned student are in this PersonalTimeTable. StatisticalInfo stores the statistical groups this students belongs to. This can be anything. In daily practice, for example, this can be a minor this student participates in. Also non-straightforward criteria could be used, like the gender of the student.

4.1.8 Statistics

Statistics is a key element in demand prediction if the information about the past or intended courses of a student lacks, which will be frequently the case (especially the latter one). Statistics is coupled to a statistical group, which is one field of the data element. The other field expresses the expected demand for that group. Per group,

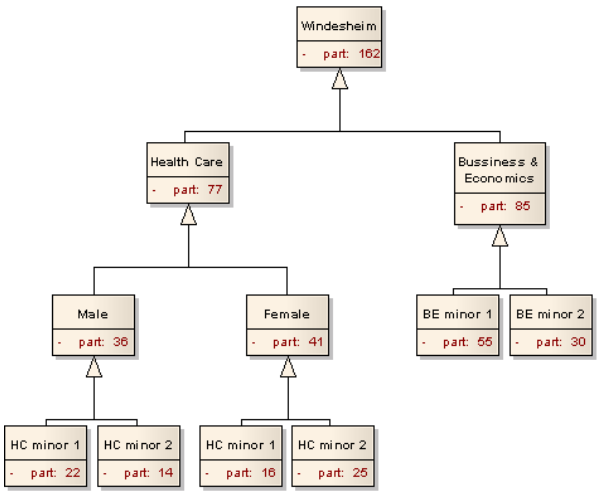


Figure 4.1: Example of a statistics tree

there is a relation to a course if there is any information on the expected amount of participants for that course. Statistics has a tree-like structure, that is expressed in the relation from statistics to itself. The root is obliged to have statistical information of all the courses that should be scheduled. A statistics element can have at most one parent. In fact, all but one will have one parent (only the root is orphan). A statistics element can have more than one child. An example statistics tree is given in figure 4.1. Every node in the statistics tree contains information based on the past and insight of domain experts, like a head master. In the statistics tree, only the node name and the expected participants for that group are shown. For example, the number of expected students for Health Care is 77. Remark that a parent contains its children exclusively, and therefore information aggregates up to the root of the tree (77 students of Health Care is the sum of its male and female students). The relation to courses is left out in figure 4.1. In figure 4.2, the node corresponding to "'BE minor 2"' of figure 4.1 is shown

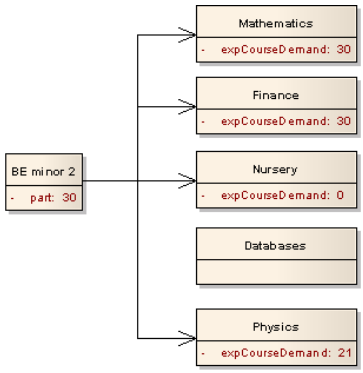


Figure 4.2: A node in the statistics tree

for a world with five courses. All the students belonging to this group participated in the courses "'Mathematics"' and "'Finance"' (typical if such a course is mandatory for the considered group). Students of this group never participate in "'Nursery"', while "'Physics"' is quite popular. The statistics for course "'Databases"' for students of group "'BE minor 2"' lacks, so no relation is defined. This lack will be filled by the closest parent of "'BE minor 2"' that has this information. It will be available somewhere, since the root is obliged to have statistics for every course. Besides total amount of participants per group and participants per course per group, the students in common with the other courses are stored per course per group. This is structured similar to figure 4.2, but instead of one entry "'expCourseDemand"' there is an entry for all the other courses, expressing the number of students that the course has in common with the other courses.

4.1.9 EduPeriod

The EduPeriod data element is a collection of course that are offered in the same education period.

4.1.10 TimeTable

The timetable data element is similar to the PersonalTimeTable of a student, as described in section 4.1.7. Difference is that this timetable contains all the lessons that take place.

4.2 Testworld

To validate a timetable engine, testdata is necessary. For this purpose, a testworld is created. This is done by combining manually created data files with files generated by a tool. In the testworld, all the data that is needed prior to timetabling is present. This section describes the testworld, by describing each of its elements. together with the elements, their origin is presented.

4.2.1 General data

As a base of the testworld, the concerned timespans must be given. Those are stated in a manually created file. An example entry is given in table 4.1. Planning cycle here expresses how many weeks the planningcycle takes. Similar, "'Education Period" expresses its duration in weeks. The last column expresses the number of plannable time units (which will commonly be hours) of a week.

Planning cycle	Education period	Time units per week
40	10	50

Table 4.1: General data

4.2.2 Teachers

Teachers in the testworld are manually created. The testworld contains 5 teachers. An example entry in the teacher table is shown in table 4.2. The row "Unavailable" indicates the hours in the week that this teacher is not available. Max. indicates the number of hours the teacher is willing to teach (which can only be at his available hours).

First Name	SurName	Id	Unavailable	Max.
Isaac	Newton	IN	1 2 3 49 50	31

Table 4.2: Teacher data

4.2.3 Rooms

Rooms of the testworld come from a slightly adapted table of the "School of Health Care". In the testworld, 8 rooms occur. Table 4.3 is an example entry of room data. The second column gives the id of the room, the last column gives the type of this room, which is used to identify lessons that can take place here.

Capacity	Room	Room type
25	E226	Practice_Hospital

Table 4.3: Room data

4.2.4 Roomstructure

Rooms have a type. In the description of a lesson, a required type in the room field can be entered. For a room to be a good candidate, the types need not to match exactly. A match with one of the parent types of the room will also do. Roomstructure is structured like shown in figure 4.3. For example, if a room for a lesson needs to be of type "Practice", it can be a room with type "Practice" or one of the subtypes (for example, the room of table 4.3 is valid to host this lesson, since its type is a subtype of the required type).

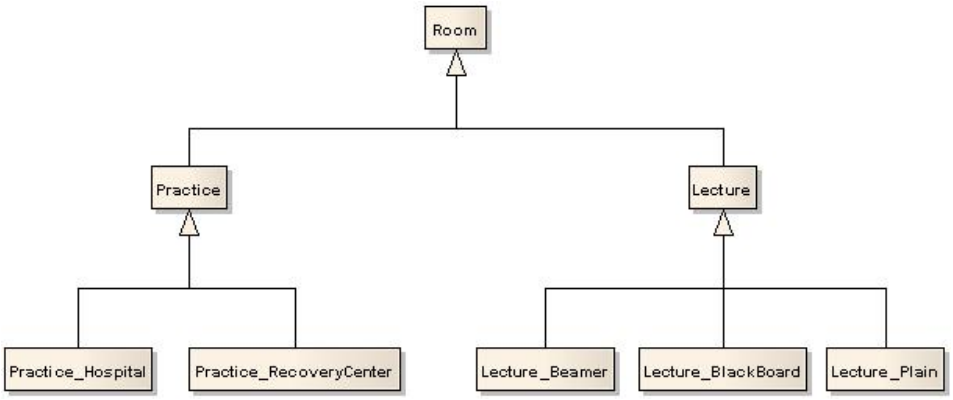


Figure 4.3: Room structure

4.2.5 Course definition

Courses

Courses consist of lessons, as stated in section 2.1. The courses that are to be scheduled are read from a table, which has rows for each course. A row describing a course is followed by rows describing its lessons, one row per lessontype. An example course describing row is printed in table 4.4. Here, the column "Min." expresses the minimum number of periods that a course should be offered (trivially, "Max." means the maximum number of periods this course is allowed to be offered). The columns "pred." and "succ." express the order in which students should take courses. The courses in the column "Pred." are the predecessors of this course, on other words, only students that have passed those courses can participate in this course. Likewise, students can only participate in the courses printed in the column"Succ." (successors) if they have passed this course. Such a precedent relation can either be defined in the predecessor column or in the successor column. In a preprocessing phase, the system automatically extends this relation. The last two columns restrict the course placement. Columns "Not with" prints the courses that can not be in the same period as this course, while "Not in per."

prints the periods in which this course could not take place. 22 Courses are generated in the testworld.

Id	Description	Min.	Max.	Pred.	Succ.	Not with	Not in per.
C1007	Articuleren	1	1	C1005 C1006		C1000	2 4

Table 4.4: Course data

Lessons

Right under the courses, the lessons of this course are printed. A course in the testworld has on average 2 types of lessons, each to be taught on average 2 hours a week. An example entry of such a lesson definition is printed in table 4.5 (note that rows and columns are switched here compared to table 4.4). The field "Hours" expresses the number of hours a student participating in the corresponding course should have this lesson in the week. Requirements to the duration of a lesson are expressed in the two succeeding fields. If the minimum duration of a lesson is two hours, this lesson is supposed to be taught back to back (as is the case for lesson "C1007_0"). Likewise, the next two fields express the bounds to the number of students for which it is acceptable to provide a lesson of this kind. Resources are needed to facilitate a lesson. Resources dealt with in this thesis are rooms and teachers. For rooms, a required type (as explained in section 4.2.4) is printed. Potential teachers are listed at the teachers field. It might occur that lesson have an order during the week (for example when a lecture must precede a practical session). This can be expressed in the last two fields and works similar to those fields for courses.

Id	C1007_0	C1007_1
Description	Articuleren, les 1	Articuleren, les 2
Hours	2	2
Min. duration	2	1
Max. duration	2	2
Min. students	10	5
Max. students	250	20
Roomtype	Lecture	Practice_RecoveryCenter
Teachers	AE AL	IN AL AE
Pred.		C1007_0
Succ.		

Table 4.5: Lesson data

Generating course definitions

Course definitions as described above are generated using a tool. This tool takes as an input:

- List of course descriptions

- List of roomtypes
- List of teachers
- Number of education periods
- Average number of class hours a course takes

Based on this information, a random course definition is generated for all the course descriptions that were put in.

4.2.6 Students

Student data

The testworld contains 400 students. For each student, the data like given in table 4.6 is available. Row "Group" defines the statistical group this student belongs to (as in 4.1.8). The fields "Past" and "Planned" state the courses that this student has already past, or is intended to participate in (which is valuable information for demand prediction). Last column gives the number of courses this students wants to participate in per education period. If this is not given, it should be a part of the demand prediction.

First Name	SurName	Id	Group	Past	Planned	Demand
Thijmen	Kin	S1148	HC_Minor3	C1021 C1005	C1018	3 3 4 3

Table 4.6: Student data

Generating student data

For the generation of student data, a tool is used. This tool uses as an input:

- List of courses with their precedence relations
- List of possible statistical groups
- List of first names
- List of surnames
- Average number of courses a student participates in per period

The courses with their precedents are read from the result of section 4.2.5. Names are read from a table with 401 different first names and 1716 different surnames. Using this information, random students are generated.

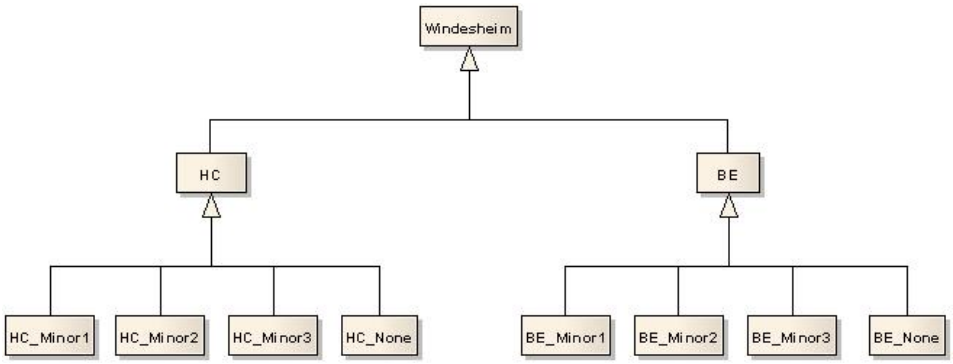


Figure 4.4: Testworld statistics

4.2.7 Statistics

Statistical data

Statistical information has a central role in demand prediction. Section 4.1.8 defines the structure. The groups and their structure used in the testworld is given in figure 4.4. This structure is put in manually. Statistical information is per group (so, per node in the tree). This information is read from a table, that is structured like table 4.7. The first row here says that the statistical information concerns the statistical group "HC_Minor1", which has an average of 30 students in the past. Below this row, a table expresses the number of students that a pair of courses has in common (for example, course "C1000" and course "C1001" had 7 students in common on average in the past). Note the pair of a course with itself expresses the number of students that participated in this course on average in the past (the main diagonal).

HC_Minor1	30				
	C1000	C1001	...	C1020	C1021
C1000	16	7	...	9	7
C1001	7	17	...	9	8
...	...	...	...	...	...
C1020	9	9	...	14	9
C1021	7	8	...	9	16

Table 4.7: Statistical data

Generating statistical data

Statistical information is generated using a tool (except for the line describing the structure). For generating the statistical information, the input used is:

- Course ids
- Average number of courses a students participates in in a year

The consumption of courses also differs per group. For example, of the offered courses, a student of the group “HC” participates in more courses than a student of the group “BE”.

Chapter 5

Quantification

This chapter describes how a score can be assigned to a given solution, at distribution level, timetable level and student matching level. How to come to a feasible solution is described in the next chapter, just as the hard constraints that define a feasible solution.

5.1 Quantifying curriculum distribution

In section 2.4.2 is defined how a distribution of the courses of the curriculum among the education periods is qualified. To compare the quality of distributions, this should be quantified. This section defines how the components are quantified. The total score for the distribution is the weighted average of those components. Important for a good distribution:

- Balanced student program
- Balanced resource usage

5.1.1 Representing a distribution

A distribution of courses among the education periods of the planning cycle is represented by storing for every education period which courses are offered. Per course is stored which lessons are part of it. Those lessons are not assigned to a specific hour, but have resources assigned to it. This is done to be able to keep track of the utilization factor of resources. So the variables here are:

C_p	The set of offered courses in education period p
$R_{c,p}$	The set of resources used for course c in education period p

Variables used in this section are most often derived of these variables.

5.1.2 Balanced student program

A balanced distribution for a student was qualified (in section 2.4.2):

- A student is able to fill the planning cycle with a maximum amount of courses related to his learning route (not a half occupied education period preceded by an education period with overlap)

To realize this objective, we want to penalize two strongly related courses taking place in the same period. How strong two courses relate can be quantified by the number of common students. However, if there are a lot of other possibilities to participate in one of these courses, we should penalize less. If two courses have a lot of overlapping students, we consider it more important to balance their distribution. The variables for determining the score are the number of education periods two courses are both offered, ϕ_{c_1,c_2}, and the number of education periods each of the courses is offered itself, ϕ_c. We formulate the penalty costs as follows:

$$F^{\text{bal}}_{\text{distr}} = \sum_{c_1 \in C} \sum_{c_2 \in C/c_1} f_{c1,c2}(\phi_{c_1,c_2}, \phi_{c_1} \cdot \phi_{c_2}) \tag{5.1}$$

$$f_{c1,c2}(x, y) = |c_1 \cap c_2| \cdot \frac{x}{y} \tag{5.2}$$

C	The courses that are offered
C/c_1	The courses that are offered, except course c_1
$\lvert c_1 \cap c_2 \rvert$	The number of overlapping students of courses c_1 and c_2

Because we consider it more important to balance if more students are involved, we multiply by the number of students that participate in both courses. The actual score for overlapping courses is the quotient of the number of periods the courses are both offered and the possible combinations of periods to participate in both courses (formula 5.2). The total score is taken for all possible combinations of courses (formula 5.1).The lower this outcome, the better. An example of the use of this formula:

Suppose we consider course c_1 and course c_2. There are 40 students participating in both. Course c_1 is offered just in education period one, as is c_2.

$$\begin{aligned} f_{c1,c2}(x, y) &= |c_1 \cap c_2| \cdot \frac{x}{y} \\ &= 40 \cdot \frac{1}{1} \\ &= 40 \end{aligned}$$

This situation gives the full overlap as penalty, which makes sense, since there is only one possibility for the students to participate in those courses, and that possibility is at the same time. A possible improvement is to offer both courses in periods one and three

instead of only period one. Assuming the available capacity will be equally distributed among the two considered periods, the penalty costs will be:

$$\begin{aligned} f_{c1,c2}(x,y) &= |c_1 \cap c_2| \cdot \frac{x}{y} \\ &= 40 \cdot \frac{2}{4} \\ &= 20 \end{aligned}$$

Parameter y is 4 here, because there are four possible ways to participate in both course 1 and course 2. The score of 20 can be further improved by, for example, move course 2 from education period 3 to education period 2. Parameter x will then be reduced to 1 (since there is only one period that both courses occur), while the other parameter remain the same. Penalty cost will then be 10.

Remark that the expected number of overlapping students is reliable if the students are randomly distributed, while in real life students will select their best option. This approach is still useful, since its use is only scoring bad situations worse that good situations. Having little overlap in random distribution means a lot of options for students, while great overlap excludes some possibilities.

5.1.3 Balanced resource usage

Resources are the entities needed to facilitate a lesson. For simplicity, the only resources considered are teachers and rooms. A balanced use of resources means:

- Resources are equally used during the planning cycle

The use of a resource can be expressed by its utilization factor (the quotient of its use and availability (maximum number of hours that it can be used)). Hence the availability must be known. If the current utilization factor is higher than a predefined perfect utilization factor, a penalty cost is assigned. For a resource with the perfect utilization factor, a decrease of its use is not considered an improvement. The variable that determines the score is the number of used hours of a resource in a particular education period, $\rho_{p,r}$ (where p gives the period and r the concerned resource). We formulate the penalty costs as follows:

$$F_{\text{distr}}^{\text{util}} = \sum_{p \in P} \sum_{r \in R} f_{p,r}(\rho_{p,r})^b \tag{5.3}$$

$$f_{p,r}(u) = \frac{\max\left(0, \frac{u}{a_{p,r}} - \widehat{\rho}_{p,r}\right)}{1 - \widehat{\rho}_{p,r}} \tag{5.4}$$

P	The education periods of the plancyle
R	The resources
$f_{p,r}(u)$	The score for the utilization factor of resource r in education period p, if used u hours
b	Utilization factor close to 1 is very undesirable. By assigning $b > 1$, this situation is penalized heavier
$a_{p,r}$	The availability of resource r in education period p
$\widehat{\rho}_{p,r}$	The perfect utilization factor of resource r in education period p

Formula 5.4 expresses how a score is assigned to the use of a resource. The base here is the difference between the actual utilization factor and the perfect utilization factor for that resource. Using a resource less when having the perfect utilization factor neither needs to be penalized nor rewarded, so the maximum of the difference and 0 is taken in the counter of formula 5.4. The resulting difference is spread over an interval of $[0 \ldots 1]$ by taking the difference of the full use (utilization factor is 1 then) and the perfect use as the denominator. If a resource is assigned an extra hour to it, this is considered less undesirable when its utilization factor is close above the perfect utilization factor than when this is close under the full use. To express this, the result of formula 5.4 is raised by the power of b in formula 5.3, with $b > 1$. As a result, adding an hour in the latter case is penalized heavier. This is done for all the resources at all education periods. If the availability of resources is not known at this stage, $a_{p,r}$ and $\widehat{\rho}_{p,r}$ are undefined. To get an equal spread, the total number of hours is taken for $a_{p,r}$ and 0 is used for $\widehat{\rho}_{p,r}$.

Formula 5.4 is plotted and shown in figure 5.1. The horizontal axis represents the factor $\frac{u}{a_{p,r}}$. Four situations are considered, differing in the perfect utilization factor. Considered perfect utilization factors are 0.6 (dark blue), 0.7 (pink), 0.8 (yellow) and 0.9 (turquoise). Three plots are made, varying in factor b. Factor b decides how increments in utilization in different situations relate to each other. In the case $b = 1$, the increment in penalty is equal, no matter the current utilization factor. By increasing b, the penalty for an increment of utilization is bigger if the current utilization factor is higher (if this is higher than the perfect utilization factor). This is the behavior that is probably desirable, since an increment from a utilization factor of 0.76 to 0.78 is in most practical cases more acceptable than an increment of 0.96 to 0.98.

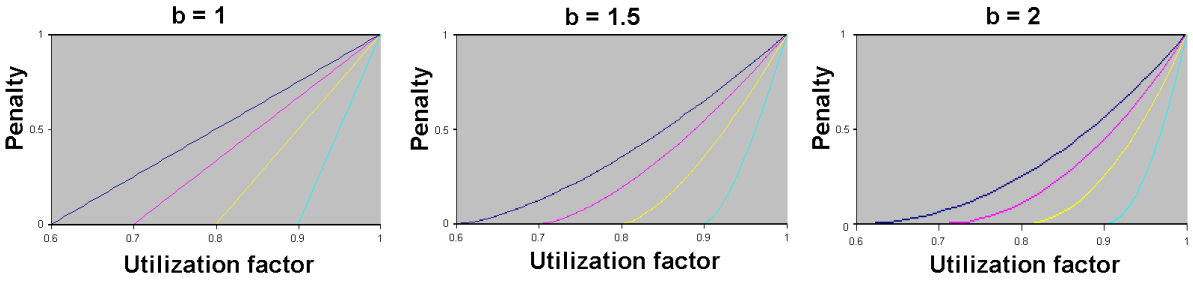


Figure 5.1: Plot of formula 5.4

5.2 Quantifying a timetable

The quality of timetable depends on a number of parameters, as defined in section 2.4.2.

5.2.1 Representing a timetable

A timetable is represented by a set of lessons for every hour in the week, and for a dummy hour. This dummy hour represents the lessons that are not taught, but can be taught in a feasible schedule (more on how this dummy hour is composed in section 6.4.2). A lesson has resources (teacher and room) attached that facilitate this lesson. So the variables here are:

L_w	The set of offered lessons at week hour w
L_D	The set lessons at the dummy hour
R_l	The set of resources used to facilitate lesson l

Variables used in this section are most often derived of these variables.

5.2.2 Idle time (student)

In section 2.4, minimization of the idle time was one of the main issues for the qualification of a time-table. Since there are no individual students coupled to lessons at timetable stage, it is not possible to sum the individual idle time over all the students. Instead, we estimate the overlap between courses in adjacent hours, giving a reward if there is a big throughput to the next hour. That can be done, based on the general expectation concerning the student subscription. This general expectation consists of a number of expected participants for every course and an estimation of the number of students that participate in a pair of courses for every pair of courses. The variables determining the score for idle time at scheduling level, are the hours in the week a lessontype is offered. This is expressed in T_w, meaning the lessontypes that are offered at hour w. We formulate the reward function as follows:

$$F_{\text{sched}}^{\text{idle\_s}} = \sum_{w \in W} \sum_{t_1 \in T_w} \sum_{t_2 \in T_{w+1}} f_w(t_1, t_2) \tag{5.5}$$

$$f_w(t_1, t_2) = \begin{cases} 0 & \text{if } w = last_w \\ 0 & \text{if } t_1 = t_2 \\ \chi_{t_1,t_2} \cdot \min\left(\dfrac{|S_{t_2,w+1}|}{|S_{t_1,w}|}, 1\right) & \text{otherwise} \end{cases} \tag{5.6}$$

W	The hours of the week that education is possible
T_w	The types of lesson that take place at hour w
$f_w(t_1, t_2)$	The score for possible throughput
$last_w$	The last hour of the day that hour w is part of
χ_{t_1,t_2}	Table expressing the overlap between lessontypes t_1 and t_2
$\lvert S_{t_1,w}\rvert$	The number of students that lessontype t_1 at hour w provides capacity for

The total reward is the sum of the rewards of all the hours, for all the possible combinations of offered lessons of the considered hour with lessons that are offered the next hour (formula 5.5). If the considered hour is the last hour of the day, we do not care about the overlap with the next hour (which will be the first hour of the next day). To prevent a reward for a sequence of equal lessons, no reward is given if the succeeding lessons are of the same type. Otherwise, a relation score for two lessontypes is read from a table (formula 5.6). The result from this table is multiplied by the part of the students that can go on to the next hour, with a maximum of 1. For example, if a lesson that is offered at a particular hour shows much overlap with some lessontype offered at the next hour, but the student capacity of that lessontype at the next hour is far less, there is still the risk of idle time.

The table expressing the overlap is a two dimensional table of $|T| \times |T|$ elements, where $|T|$ is the total number of different lessontypes. A cell in the table is an integer number within the interval of $[0 \dots x]$, where x can be chosen arbitrarily. The cell t_1, t_2 expresses how many sets of students of t_1 also participate in t_2, if t_1 is divided in x equal sets. Remark that by this definition, cell t_1, t_2 is not necessarily equal to cell t_2, t_1. A student is able to subscribe for an entire course only, subscription for lessontype is not possible. Because of this, $\chi_{t_1,t_2} = \chi_{t_3,t_4}$ if t_1 and t_3 belong to the same course, and t_2 and t_4 belong to the same course.

Suppose there are 50 participants in course c_1, and 60 participants in course c_2. There are 27 students participating in both. Course c_1 consists of one "'lecture"' hour and one "'college"' hour, as is the case with course c_2. Now suppose at hour 5 a lesson of c_1 with type "'college"' takes place having 30 participants, and at hour 6 a lesson of c_2 with type "'college"' takes place having 20 participants. These are the only lessons that take place that day. A day consists of 10 hours here. Looking at formula 5.5, there is only one combination for which formula 5.6 needs to be applied. That is, $f_5(t_{\mathrm{c}_1,\mathrm{college}}, t_{\mathrm{c}_2,\mathrm{college}})$. Since 5 is not the last hour of the day, the score will be read from the table. Suppose we took 4 as the number of segments the overlapping students are placed in. The total number of students for lessontype $t_{\mathrm{c}_1,\mathrm{college}}$ will be 50. Each part will have a size of $\frac{50}{4} = 12.5$. Since 27 students overlap between $t_{\mathrm{c}_1,\mathrm{college}}$ and $t_{\mathrm{c}_2,\mathrm{college}}$ and this is in the third segment (from 25 to 37.5), $\chi_{(t_{\mathrm{c}_1,\mathrm{college}}),(t_{\mathrm{c}_2,\mathrm{college}})}$ will contain the value 2 (since 0 and 1 are assigned for the first two segments). This will be multiplied by $\min\left(\frac{|S_{w+1}|}{|S_w|}, 1\right) = \min\left(\frac{20}{30}, 1\right) = \frac{2}{3}$. Hence, the returned score to formula 5.5 will be $2 \cdot \frac{2}{3} = 1\frac{1}{3}$.

5.2.3 Resource utilization factor

Like in section 5.1.3, we want a balanced use of the available resources. The formula expressing this (formula 5.7), is almost identical to formula 5.3. Only difference are some terms concerning time. Instead of using periods as time units (in formula 5.3), hours of the week are used here. However, this does not change the formula. Because of this similarity, we do not discuss the formula in full detail here. For a detailed discussion, we refer to section 5.1.3. The slightly adapted penalty function:

$$F_{\text{sched}}^{\text{util}} = \sum_{r \in R} f_r(\rho_r)^b \tag{5.7}$$

$$f_r(u) = \frac{\max\left(0, \frac{u}{a_r} - \widehat{\rho}_r\right)}{1 - \widehat{\rho}_r} \tag{5.8}$$

R	The resources
$f_r(u)$	The score for the utilization factor of resource r, if used u hours
b	Use increase close to full use is penalized harder if $b > 1$
a_r	The hours resource r is hired for
$\widehat{\rho}_r$	The perfect utilization factor of resource r

5.2.4 Travel time

If traveling to the next lesson takes to much time for students, this can be a serious disturbance of this next lesson. Therefore, we want to penalize travel time above a certain threshold. This threshold depends of the hour transition; if there is a big break after an hour, the threshold can be higher than when the next hour immediately succeeds this hour. Further more, if lessons have a lot of students in common, travel time is considered more important. The variables deciding the travel time score are the hour a lesson is offered and the location the lesson is offered. This is expressed in L_w, meaning the lessons that are offered at hour w. We formulated the penalty for travel time as follows:

$$F_{\text{sched}}^{\text{travel}} = \sum_{w \in W} \sum_{l_1 \in L_w} \sum_{l_2 \in L_{w+1}} g_w(l_1, l_2) \tag{5.9}$$

$$g_w(l_1, l_2) = \begin{cases} 0 & \text{if } \psi_{l_1,l_2} \leq \Psi_w \\ \chi_{t_{l_1},t_{l_2}} \cdot (\psi_{l_1,l_2} - \Psi_w) & \text{otherwise} \end{cases} \tag{5.10}$$

W	The hours of the week that education is possible
$g_w(l_1, l_2)$	The penalty for travel time from l_1, offered at hour w, to l_2, offered at hour $w+1$
ψ_{l_1,l_2}	The travel time needed to travel from the location of l_1 to the location of l_2
Ψ_w	The threshold for travel time at the hour transition of w to $w+1$
χ_{t_1,t_2}	Table expressing the overlap between lessontypes t_1 and t_2

The total penalty for travel time is the sum of all the hours and all the combinations of lessons of that hour with the next hour (formula 5.9). Formula 5.10 returns a penalty of 0 if the travel time is below the threshold, and the travel time above the threshold multiplied by their overlap. This overlap is expressed in a table identical to the table formulated in section 5.2.2.

5.2.5 Overlapping courses

Lessons that have a lot of students in common are preferably not offered at the same hour. To realize this, we have a penalty function for related lessons at the same hour. The ideas of this function much resemble the ideas of section 5.2.2. Only instead of rewarding related back to back lessons, we penalize related lessons at the same hour. The variables determining the penalty for overlap at scheduling level, are the hours in the week a lessontype is offered. This is expressed in T_w, meaning the lessontypes that are offered at hour w. We formulate the penalty function as follows:

$$F_{\text{sched}}^{\text{overlap}} = \sum_{w \in W} \sum_{t_1 \in T_w} \sum_{t_2 \in T_w / t_1} h_w(t_1, t_2) \tag{5.11}$$

$$h_w(t_1, t_2) = \chi_{t_1,t_2} \tag{5.12}$$

W	The hours of the week that education is possible
$h_w(t_1, t_2)$	The penalty for overlap of lessontype t_1 to lessontype t_2
χ_{t_1,t_2}	Table expressing the overlap between lessontypes t_1 and t_2

The total penalty is computed by taking the sum over all hours and all combinations of lessontypes that occur at that particular hour (formula 5.11). Table χ_{t_1,t_2} is identical to the table formulated in section 5.2.2.

5.2.6 The order of lessons

In some occasions, it may be mandatory for a students to participate in a certain lesson in the week before participating in another, for example, when he must participate in a practical session after participating in a related lecture. However, this does not mean that it is mandatory for every instance of a practical session to be taught after the last instance of lecture (only that the first of all occurrences of both is a lecture and the last is a practical session). To quantify this, we assign a reward for the part of the net capacity for a succeeding lessontype that is offered after the corresponding preceding lessontype. The variable determining the score for the order of lessons is the net number of students of a succeeding lessontype that takes place after the lesson it is a successor for, expressed in $|S_{l+,t}|$. Here, $l+$ expresses the hours after lesson l took place and t expresses the succeeding lessontype. We formulate this reward as follows:

$$F_{\text{sched}}^{\text{order}} = \sum_{l \in L} \sum_{t \in T_{l-}} f_t(|S_{l+,t}|) \tag{5.13}$$

$$f_t(x) = \sqrt{\frac{x}{|S_t|}} \tag{5.14}$$

L	The lessons that take place at hour w
T_{l-}	The lessontypes that are predecessors for lesson l
$f_t(x)$	Function expressing the order reward for a lesson that has to precede lessons of type t, if a net capacity of x of type t is offered after this preceding lesson
$\|S_t\|$	The total net capacity for students of lessontype t

This reward is computed for all lessons in the week that have succeeding lessontypes (formula 5.13). The reward itself is computed by taking the part of the capacity of the succeeding type that actually succeeds this lesson. Taking the square root is done to make the reward more sensible to changes if only a small part of the capacity of a succeeding lessontype actually succeeds the lesson.

5.2.7 Division

It is not desirable to have all lessons of a certain type at the same day. For this, we want to penalize deviations from the prefect time spent on a lessontype for a day. The variable determining the penalty for this, is the number of hours that a lessontype is provided per day, expressed in $|L_{t,d}|$, where t is the lessontype and d the day. We formulate the penalty cost as follows:

$$F_{\text{sched}}^{\text{div}} = \sum_{t \in T} \sum_{d \in D} g_{t,d}(|L_{t,d}|) \tag{5.15}$$

$$g_{t,d}(x) = \frac{|x - |\widehat{L}_{t,d}||}{|L_t|} \tag{5.16}$$

T	The lessontypes
D	The days of the week
$g_{t,d}(x)$	Function returning the penalty for the deviation of lessontype t at day d, if t is taught x hours at day d
$\|\widehat{L}_{t,d}\|$	The preferred number of hours for lessontype t to take place at day d
$\|L_t\|$	The total number of hours that lessons of type t are offered

The total penalty is the sum over all lessontypes and all days(formula 5.15). Per lessontype and per day, this penalty is computed as the deviation of the preferred hours relative to the total number of hours (formula 5.16).

5.2.8 Idle time (teacher)

The qualification of the idle time for teachers was qualified:

- Not to little idle time for teachers

To quantify this, we penalize the situation in which a teacher has more of less idle time than he prefers for a day. The variable determining the penalty for this deviation

is the number of actual idle time per day, $\iota_{t,d}$, expressing the idle time that teacher t has at day d. We formulate the penalty cost as follows:

$$F_{\text{sched}}^{\text{idle\_t}} = \sum_{te \in Te} \sum_{d \in D_{te}} h_{te,d}(\iota_{te,d}) \tag{5.17}$$

$$h_{te,d}(\iota_{te,d}) = |\widehat{\iota}_{te,d} - \iota_{te,d}| \tag{5.18}$$

Te	The teachers
D_{te}	The days teacher te has availability > 0
$h_{te,d}(\iota_{te,d})$	Function returning the penalty for idle time of teacher te at day d
$\widehat{\iota}_{te,d}$	The idle time teacher te prefers at day d

The total penalty is the sum of all the deviations of all the teachers at the days they are available(formula 5.17). Formula 5.17 gives the deviation from the preferred number of idle time for a teacher at a specific day.

5.2.9 Student preferences

Acknowledging the preferences of the students contributes to a good timetable. Student preferences concern preferred time and courses they want to participate in. These course preferences are used for a more reliable estimation of the course demand, and are not further considered here. The acknowledgment of the time preference of the students is taken into account. A reward is assigned if a lessontype is offered at a time the students of that lessontype prefer. The variable determining the reward is the moment a lessontype is offered, expressed in $g_t(w)$, with w being an element of W, being the hours of the week.

$$F_{\text{sched}}^{\text{pref\_s}} = \sum_{w \in W} \sum_{t \in T} g_t(w) \tag{5.19}$$

$$g_t(w) = |S_{w,t}| \cdot \tau_{w,t} \tag{5.20}$$

T	The lessontypes
$\lvert S_{w,t}\rvert$	The net capacity for students of lessontype t at time w
$\tau_{w,t}$	The popularity of hour w for lessontype t to take place

The reward for time preference is summed over all hours of the week for all lessontypes (formula 5.19). Net capacity of a lessontype of an hour is the number of students that can participate in that course at that hour at max. This is defined as the minimum of the students places for that lessontype at that hour and the total participants for that lessontype. Popularity of an hour for a lessontype is computed by averaging the popularity for that hour of the students that are expected to participate in that lessontype (to keep it simple, a possibility is to express this in only 2 values per student, for example 1 (not popular) and 2 (popular)).

In real life, students will probably not be asked to give their preference at hour level. Reason for this is that this is hard to acknowledge for every student, which will result in grumpy students. Instead, preference for a collection of hours (for example: morning) can be asked, which will be easier to acknowledge and hence results in more satisfied students.

5.2.10 Teacher preferences

To quantify how much the teacher preferences are acknowledged, the teacher preferences themselves need to be quantified first. This is done by assigning a preferred time ratio a teacher would teach the lessontypes within his qualification.

Expected time ratio

The expected time ratio that a teacher spends on a course depends on the availability of the teachers, the required teaching hours for each lessontype and the qualification of the teachers. From this input data, for each teacher a list is created that has an entry for each lessontype, containing the ratio that divides the available hours of the teacher by the available capacity for the lessontype. The capacity of a lessontype is the sum of the availabilities of the teachers that are qualified to teach that lessontype. For example, suppose there are four teachers $te_1 \dots te_4$, each of them available ten hours a week. Further, there are four lessontypes $t_1 \dots t_4$, each to be taught ten hours a week. Qualification of teachers is according to table 5.1.

Teacher	Qualification
te_1	t_1, t_2, t_3
te_2	t_2, t_4
te_3	t_1, t_3
te_4	t_2, t_3

Table 5.1: Qualification

In the example, t_3 has a capacity of thirty hours, since it can be taught by three teachers with a total capacity of thirty hours. The resulting ratios are given in table 5.2. Iteratively, one hour of a lessontype is assigned to a teacher for the highest correspond-

	t_1	t_2	t_3	t_4
te_1	$\frac{10}{20}$	$\frac{10}{30}$	$\frac{10}{30}$	0
te_2	0	$\frac{10}{30}$	0	$\frac{10}{10}$
te_3	$\frac{10}{20}$	0	$\frac{10}{30}$	0
te_4	0	$\frac{10}{30}$	$\frac{10}{30}$	0

Table 5.2: Ratios

ing teacher/lessontype ratio and the ratios are updated. If all the hours are assigned, the expected number of hours that a teacher spends on a lessontype is known, for all lesssontypes. The expected ratio is computed by dividing these values by the number of hours a teacher is available.

Preferences

Now, suppose that teacher te_1 likes to teach lessontype t_1. Suppose that the expected ratio teacher te_1 spends on lessontype t_1 is four out of ten hours. This ratio is now adapted by multiplying the counter by a factor p, for example 1.25. The denominators of all the lessontypes that are taught by this teacher are increased with the amount that the counter is increased. So, in this case, the counter of the preferred course is increased by one and becomes five, and the denominators of all the ratios belonging to this student are increased by one and become eleven.

A situation might occur in which a popularized lessontype has a higher ratio than a lessontype that has to be taught by the concerned teacher to get a feasible solution. This does not get us into trouble, since the table with adapted ratios is only used to compute the penalty in a given timetable. However, this timetable will not be infeasible, since the algorithm will not allow the application of operators causing infeasibility. We can conclude that such a timetable will never get a penalty of 0.

The adapted parameters are taken into consideration while quantifying how much the teacher preferences are acknowledged, by comparing the ideal situation to the actual situation. Variable determining the score for teacher preferences is the ratio of hours that a teacher te teaches the lessontypes with modified preferred ratios and his available hours, expressed in θ_{te}. The ratio of teacher te teaching lessontype t (with modified preferred ratio) is referred to as $\theta_{te,t}$. We formulate the penalty cost for deviation of teacher preferences as follows:

$$F_{\text{sched}}^{\text{pref\_t}} = \sum_{te \in Te} h_{\text{te}}(\theta_{te}) \tag{5.21}$$

$$h_{\text{te}}(\theta_{te}) = \sqrt{\frac{1}{|te_T|} \sum_{t \in te_T} \left(\theta_{te,t} - \widehat{\theta}_{te,t}\right)^2} \tag{5.22}$$

Te	The teachers
$h_{\text{te}}(\theta_{te})$	Function expressing the penalty cost for teacher te spending θ_{te} hours to teaching lessontype t
te_T	The lessontypes within the qualification of teacher te that have a modified preferred ratio
$\|te_T\|$	The number of elements in te_T
$\widehat{\theta}_{te,t}$	The preferred ratio of teacher te for $\theta_{te,t}$

Total penalty is the sum of all the penalties for individual teachers (formula 5.21). These individual penalties are computed by comparing the actual ratio to the preferred ratio of teaching a lessontype. Comparing is done in a way similar to calculating the standard deviation, only instead of the mean the preferred ratio is taken. The ratios are only compared if they have their preferability changed. The reason for the other lessontype to be not considered is not only for efficiency purpose, but also to prevent

courses the teacher is indifferent about from forming needles constraints. Considering them would result in a reward for a (close to) uniform distribution, which could compensate an unwanted choice somewhere else. In the worst case, every teacher has a deviation from his preferred ratio of 1 for all the lessontypes he has entered a preference for.

5.3 Matching students

If a good timetable according to the preceded conditions is found, one step is to be made before this potential good timetable can be considered a good timetable. Students are coupled to lessons, and the sum of the individual scores for students examines how good this timetable is for the students. A qualification for a good students matching was defined in section 2.4.2. A matching of the students to the lessons is quantified by taking a weighted average over the quantifications of those qualifications.

5.3.1 Representing a student matching

A student matching is represented by a set of students attached to every lesson of a given timetable (will not be changed at this stage). So the variables here are:

S_l	The set of students matched to lesson l

Variables used in this section are most often derived of these variables.

5.3.2 Actual idle time

As said before, idle time is very undesirable. At the matching stage, we can simply count the idle time for every student, and use that as a penalty. The variable determining the number of idle hours is the schedule of the student, D_s, expressing the set of days in which student s has classes. we formulate the penalty costs as follows:

$$F_{\text{match}}^{\text{idle}} = \sum_{s \in S} f(D_s) \tag{5.23}$$

$$f(D_s) = \sum_{d \in D_s} (\omega_d - \alpha_d + 1) \tag{5.24}$$

S	The students
ω_d	The last hour that a student has lessons at day d
α_d	The first hour that a student has lessons at day d

The actual idle time in a week is equal to the active hours of that student in the week, decreased with the number of hours that the student has participated in lessons in the week. Active hours are counted by taking for each day the hours from the first hour the students has lessons (inclusive) until the last hour a students has classes (inclusive), as

can be seen in formula 5.24. To get the actual time, the total number of class hours should be subtracted from this. Since this value does not change at this stage, it is of no influence and we leave it behind. The total penalty is computed by summing over all students (formula 5.23).

5.3.3 Met time preferences

Students probably have preferences for which hours they would like to have their lessons. We assign a reward here that measure how much the students preferences are acknowledged. The variable determining this reward is the schedule for this student, W_s, expressing what lessons student s has at the hours of the week. We formulate this reward function as follows:

$$F_{\text{match}}^{\text{time\_pref}} = \sum_{s \in S} \frac{k_s(W_s)}{\widehat{k}_s} \tag{5.25}$$

$$k_s(W_s) = \sum_{w \in W_s} R_w \cdot b_{s,w} \tag{5.26}$$

S	The students
$k_s(W_s)$	The reward for the schedule of student s concerning the time preferability of the lessons
$\widehat{k}_s$	The perfect reward for student s
R_w	The reward for hour w
$b_{s,w}$	This variable gives 0 if student s has no education at hour w and 1 if he has

In preprocessing, for each student is computed what the maximum reward for hour preferability is. The current reward is computed in formula 5.26 by summing the reward for the hours that student s has lessons. This total reward for this student is weighted to the maximum possible reward (formula 5.25). This is done for all students.

5.3.4 Preferable day length

Students have a preferable length of the day. We use a penalty function here that penalizes the deviation from this preferred day length. The variables determining the penalty are the day schedules of student s, D_s, expressing the days that student s has lessons. We formulate the penalty costs as follows:

$$F_{\text{match}}^{\text{daylength}} = \sum_{s \in S} \sum_{d \in D_s} g(d) \tag{5.27}$$

$$g(d) = \left(|d| - \widehat{|d|}\right)^2 \tag{5.28}$$

S	The students
$g(d)$	The penalty for student s for deviation from the preferred day length at day d
$\|d\|$	The length of day d
$\widehat{\|d\|}$	The preferred length for day d

Here, formula 5.28 gives a penalty for deviation from the perfect day length. The square is taken to penalize bigger deviation harder than linear (and at the same time, no negative penalty (reward) occurs if the days is shorter than preferred). These penalties are computed for all students and all days these students have lessons (formula 5.27).

5.3.5 Travel time

To much travel time is undesirable at some hour transitions. We penalize the travel time above a certain threshold. The variables determining the penalty for travel time are the pairs of succeeding lessons, $(l_{s,w}, l_{s,w+1})$, where $l_{s,w}$ is the lesson of student s at hour w. We formulate the penalty costs as follows:

$$F_{\text{match}}^{\text{daylength}} = \sum_{s \in S} \sum_{w \in W} f_w(l_{s,w}, l_{s,w+1}) \tag{5.29}$$

$$f_w(l_1, l_2) = \begin{cases} 0 & \text{if } \psi_{l_1,l_2} \leq \Psi_w \\ \chi_{t_{l_1},t_{l_2}} \cdot (\psi_{l_1,l_2} - \Psi_w) & \text{otherwise} \end{cases} \tag{5.30}$$

S	The students
W	The hours of the week
$f_w(l_1, l_2)$	The penalty for travel time from l_1, offered at hour w, to l_2, offered at hour $w+1$
ψ_{l_1,l_2}	The travel time needed to travel from the location of l_1 to the location of l_2
Ψ_w	The threshold for travel time at the hour transition of w to $w+1$
χ_{t_1,t_2}	Table expressing the overlap between lessontypes t_1 and t_2

Formula 5.30 is identical to formula 5.10. If a student s does not have lessons at hour w or hour $w+1$, a travel time of 0 is taken. The total penalty is the sum of all the students and all their travel time (formula 5.29).

5.3.6 Group size

Although the base idea is to match the individual education demand of a student, it is possible that some lessons require group participation. For example, students need to participate 2 hours a lesson of type "project", that is offered 8 hours a week. A project needs to be done in groups of size 4 to 6. Now if 2 lessons of this type have only 1 overlapping student, it will be impossible for this students to form a group. Therefore, during student matching, students are matched such that groups can be formed. the resulting matching gives insight in whether or not there is a possibility for the students to form a group, and gives a suggestion on how to subscribe in order to realize those

possible groups. A penalty is assigned for having to much or to little students overlapping 2 lessons. The variable determining the penalty is the number of overlapping students of two lessons that are coupled, $|l_1 \cap l_2|$, where l_1 and l_2 are two coupled lessons. We formulate the penalty function as follows:

$$F_{\text{match}}^{\text{group}} = \sum_{l_1 \in L} \sum_{l_2 \in L_{l_1}} k_{l_1,l_2}(|l_1 \cap l_2|) \tag{5.31}$$

$$k_{l_1,l_2}(x) = \begin{cases} 0 & \text{if } \left\lfloor \frac{x}{\mu_{l_1,l_2}} \right\rfloor \neq \left\lfloor \frac{x}{\lambda_{l_1,l_2}} \right\rfloor \\ 0 & \text{if } \widetilde{r}(x, \mu_{l_1,l_2}) = 0 \\ 0 & \text{if } \widetilde{r}(x, \lambda_{l_1,l_2}) = 0 \\ \min\left(\mu_{l_1,l_2} - \widetilde{r}(x, \mu_{l_1,l_2}), \lambda_{l_1,l_2} - \widetilde{r}(x, \lambda_{l_1,l_2})\right) & \text{otherwise} \end{cases} \tag{5.32}$$

L	The lessons
L_{l_1}	The lessons having a group requirement with lesson l_1
$k_{l_1,l_2}(x)$	Function expressing the penalty if x participate in l_1 and l_2
μ_{l_1,l_2}	The upper bound for groups of students for lessons l_1 and l_2
λ_{l_1,l_2}	The lower bound for groups of students for lessons l_1 and l_2
$\widetilde{r}(x,y)$	The remainder after a division of x by y

The penalty function is summed over all the lessons that have group requirements (formula 5.31). There is no penalty if those lessons have a number of overlapping students for which one of the following holds:

- The number is an integer multiple of the lower bound for the group size
- The number is an integer multiple of the upper bound for the group size
- From the number, a different number of groups can composed using the lower bound as a group size compared to using the upper bound as a group size

If all these cases do not hold, no groups can be made such that every student participates in a group of the required size. A penalty term is assigned that has the size of the number of students that should be added to this couple of lessons to make it possible to compose groups within the requirements.

Chapter 6

Local Search

In the project, local search is used as optimization technique. This chapter gives details about local search in general as well as the way it is used in the project. For each level at which local search is applied (distribution, timetable, matching), a section describes its details.

6.1 Local search basics

This section will start with the explanation of the general concept of local search. After this, a local search technique is explained. This local search technique, simulated annealing, is used in this project.

6.1.1 General local search

The first step in local search is to define an initial solution. One approach is to require this solution to be feasible with respect to the hard constraint of the model (quality is not taken into account here). Another approach is to define an initial solution that is not feasible, and assign a huge penalty to every violation (huge compared to the penalty of violation of soft constraints). From this initial solution a search for improvement is started. This is done by executing a number of runs of the following receipt:

1. Find a solution y closely related to the current solution (neighbor). This is done by slightly adapting the current solution
2. Decide to accept or reject solution y
3. If y is accepted, y becomes the current solution. Go to step 1, unless a stop condition is met.

Finding a neighbor solution is problem specific and done by applying an operator. An operator slightly changes the current solution. Deciding if a solution is accepted differs per local search technique. Stop conditions can be a number of runs done, a reached score or elapsed time. If a stop condition is reached, the local search can be restarted. The initial solution for the next local search is for example the current best solution with a major change.

6.1.2 Simulated annealing

The local search technique used here is simulated annealing (as presented in [KGV83]). Simulated annealing always accepts a neighbor solution if this solution is not worse than the current solution. If a neighbor solution is worse, this solution is accepted by chance. This is done until a stop condition is met. The best solution is stored, since the current solution is not necessarily the best. A run in simulated annealing:

1. Select neighbor y from the neighbor of the current solution x and compute the cost of y, $f(y)$.
2. If $f(y) \leq f(x)$ (cost of y are not higher), then $x \leftarrow y$ (y is new current solution). else, $x \leftarrow y$ with chance p
   - $p = exp\left(\frac{f(x)-f(y)}{T}\right)$
   - Parameter T is the control parameter. At the start of the process, the value of T is such, that approximately half of the worse solutions are accepted (rule of thumb).
3. Adapt parameter T if necessary.
   - Every Q runs T is decreased by multiplying this parameter by α, with $0,9 < \alpha < 1$.
   - Parameter Q is approximately equal to 8 (rule of thumb) times the size of the neighbor space (where the neighbor space is the set of all neighbors).
4. Stop if a stop condition is met, else go to step 1.

6.1.3 Run transition

If a stop condition for a run is met, the local search is restarted. The start of the next run is an adaptation of one of the x best solutions found so far. This solution is adapted by applying the local search operators a number of times.

6.1.4 Stop conditions

There are stop conditions at two levels, at run level and local search level. A new run is no longer started if less than a given factor of runs is an accepted impairment. Restarts of the local search are done as long as at least a given factor of restarts resulted in an improvement. Sometimes it might be desirable to bound this also by time.

6.2 Motivation local search

We decided to use local search to solve the problem. Local search is applied at three levels:

- Course distribution level, where courses of the planning cycle are distributed among the education periods
- Timetable level, where a general week timetable is optimized. Students are not taken into account as individuals, but are taken into account by the expected number of students that participates in a course.
- Student matching level, where individual students are matched to the timetable.

At all these level, no algorithm that returns the optimum in polynomial time probably exists. This will be discussed in further detail in the next sections. Since no fast algorithm for an exact solution is available, we should use approximation algorithms or local search. We choose to use local search. At the start of the project, the definition of the problem was unclear and the situation new. This could likely result in adaptations to the current idea being necessary along the way. Apart from this, Educator BV strongly requested to have a working prototype at the end of the project. Since, compared to approximation algorithms, local search based algorithms can be developed faster and adapted easier later on, we decided to use local search for this project.

6.3 Course distribution level

This section describes the hard constraints for a distribution of the courses among education periods, and gives an algorithm to get a feasible solution. To conclude, operators are formulated to slightly modify a current solution. Recall that the way of representing a course distribution was formulated in section 5.1.1.

6.3.1 Hard constrains for distribution

A distribution is feasible if it respects the hard constraints. Courses can have a unary hard constraint that says that course c can not take place in period p. Two courses can have the binary constraint that they cannot be taught together. This constraint occurs in two ways:

- Course c_1 and c_2 cannot be taught together and they have a precedence relation.
- Course c_1 and c_2 cannot be taught together, order is not relevant.

Hard constraints for resources are:

- A teacher does not teach more hours then he is hired for.

- A teacher only teaches lessons that he is qualified for.
- A room is not used more hours than hired.
- A lesson is only scheduled in a rooms that is appropriate for that lesson.

6.3.2 Finding a feasible distribution

To find a feasible solution, the following is done (starting with the first education period $p = 1$):

1. Set the tail of a course (which is defined as the minimum length of the succeeding chain of courses) for all courses to 1+ the maximum tail of all its successors. Assign 0 if no successors. Assign "unknown" if not all the tail lengths are known yet.
2. If there are still courses for which the chain length is unknown, go to step 1.
3. Assign unassigned courses that do not have preceding courses to education period p, if there is no unary constraint (course c cannot take place at period p) preventing this course to take place in this period. If there are courses that are not allowed to be taught at the same time, assign the course with the largest tail. Resources needed to facilitate the course are assigned, in order of increasing utilization factor. If there are not enough resources for this course, unassign it from this period.
4. Of the remaining courses, further ignore precedent relations with courses assigned in step 3.
5. Take $p = p + 1$.
    - If p is not bigger than the number of periods and there are still unassigned courses, go to step 3.

If this terminates, and there are still unassigned courses, we take the binary constraints (precedent relations/not at the same time) no longer as hard constraints. In further matching of students, we take these constraints into account in scoring the distribution.

A possible approach if this does not provide us a feasible distribution, is to perform backtracking on the most recent choice concerning courses that were not allowed to take place at the same time. However, this gives us a running time that is exponential to the number of backtracking possibilities.

Complexity

We concluded that adding backtracking to get an exact solution gives us exponential running time. This is unavoidable if we want an exact solution, since this problem is NP-complete. The proof for this is given by a reduction from graph-coloring, as seen in [Lew07]. The graph-coloring problem questions whether or not, given a set of vertices and a set of connecting edges, there is a way to assign each of the vertices one of the k available colors in such a way that non of the vertices has a color assigned that equals the color of one of its neighbors. The proof:

We want to know if we can color a given graph with at most k colors. To find out, we create an instance of the course distribution algorithm with input:

- For every vertex v of the graph, we add a course c
- For every edge between vertices v and w, we add the constraint that the corresponding courses cannot be taught at the same period
- The course distribution algorithm is asked for a course distribution over k periods
- Of the returned solution, we assign the same color to every vertex with their corresponding course in the same period. Every period assigns their vertices a unique color.

The above shows that a problem instance of the NP-complete graph-coloring problem (a nice introduction into the theory of NP-class problems can be found in [CLRS01]) can be translated in polynomial time to an instance of the course distribution problem, and a solution of the course distribution problem can be translated in polynomial time to a solution of graph-coloring. Therefore this problem is NP-hard. Since it is trivial to decide whether a given course distribution is feasible, the problem of finding a feasible course distribution is NP-complete as well. So, unless $P = NP$, no polynomially bounded algorithm exists that is guaranteed to return a feasible solution for the general course distribution.

Assigning resources to the courses can be done in polynomial time if for every lesson, the times a qualified resource should be assigned to fulfill the demand does not depend on the resource. For example, a certain lecture needs to be provided 6 hours in a week. The qualified resources of which one is needed per provided hour are “Isaac” and “Albert”. No matter how the resources are matched to the lessons, it needs to be offered 6 times. If this is the case, the problem is a matching problem, and can be solved by solving the corresponding flow problem.

Special case

The binary constraint of courses not allowed to be taught together without a precedent relation, is constraint that makes the above reduction applicable. If this does not occur, the algorithm in section 6.3.2 will find a solution in polynomial time to the problem where we only take the *order* into account, if one exists. To find an entire feasible solution in polynomial time, the number of needed moments a lesson is offered needs to be independent of the resources used, like described in section 6.3.2.

6.3.3 Operators

The local search operators that are applied are:

Move	Move course to another education period.
Add	Offer an extra occurrence of a course.
Remove	Remove a course out of a period.
Change resource	Change a resource (teacher, room) of a lesson (lesson is invisible, but considered to keep track of the utilization factor of resources).

An operator is only applied if it does not result in an infeasible solution.

6.4 Timetable level

This section describes the hard constraints for a timetable, and gives an algorithm to get a feasible solution. To conclude, operators are formulated to slightly modify a current solution. Recall that the way of representing a timetable was formulated in section 5.2.1.

6.4.1 Hard constraints for a timetable

The first that needs to be done, is to find a feasible timetable (fulfilling the hard constraints), that will be the start solution for the local search algorithm. The hard constraints concerned here are:

- A teacher does not teach more hours then he is hired for.
- A teacher only teaches lessons that he is qualified for.
- A teacher does not teach if he is not available.
- A room is not used more hours than hired.
- A lesson is only scheduled in a rooms that is appropriate for that lesson.
- A room is not used if not available.
- A lesson is not taught if it is not available (as defined in the definition of this lesson).
- There are enough student places for every lessontype to fulfill the demand.
- Minimum time for a lesson is obeyed (back to back hours).

For the resources, there are constraints concerning number of hours and availability. The difference lies in the fact that a resource can be used n hours to facilitate a lesson, but the resource is more than n hours available for this. For example, teacher Isaac is willing to teach 20 hours and he is willing to do that at all days but Friday (availability). Now the constraint concerning number of hours makes sure Isaac does not teach more than 20 hours in total, while the constraint concerning availability makes sure Isaac does not teach at Fridays. For lessons, an availability can be entered likewise.

6.4.2 Finding a feasible timetable

Finding a feasible timetable is done similar to finding a feasible distribution (section 6.3.2). Starting with the first hour of the week, $w = 1$, this is done:

1. Generate the biggest possible set of lessons. This can be done by, for every lesson-type, calculating the number of lessons needed if all the lessons contain the minimum number of students that is allowed for a lesson of that type.
2. Split this set in the smallest possible set of lessons, such that lessons of this type can be provided for all the students willing to do that, and a remainder.
3. Assign the remainder to the dummy hour.
4. Set the tail of a lesson (which is defined as the minimum length of the succeeding chain of lessons) for all lessons to 1+ the maximum of this tail of all its successors. Assign 0 if no successors. Assign "unknown" if not all the tails are known yet.
5. If there are still lessons for which the tail is unknown, go to step 4.
6. Assign unassigned lessons that do not have predecessors to hour w, if there is no unary constraint (lessons l cannot take place at hour w) preventing this lesson to take place at this hour. If there are lessons that are not allowed to be taught at the same time, assign the lesson with the largest "minimum length of the succeeding chain of lesson".
7. Assign rooms to the lessons that are assigned to hour w. If there is not enough room capacity at this hour, unassign the lesson from this hour.
8. Assign teachers to the lessons that are assigned to hour w. If there is not enough teacher capacity at this hour, unassign the lesson from this hour.
9. Of the unassigned lessons, further ignore precedent relations with the assigned lessons.
10. Take $w = w + 1$.
    - If w is not bigger than the number of hours in the week and there are still unassigned lessons, go to step 6.

If this algorithm terminates, and there are still unassigned lessons, we take the binary constraints (precedent relations/not at the same time) no longer as hard constraints. In further matching of students, we take these constraints into account in scoring the timetable.

Complexity

This problem is an extension of the "professors and classes" timetabling problem [ECF97], which is known to be NP-complete. This problem is defined:

For a certain school with N_p professors, N_q classes, N_x classrooms and lecture halls, and N_s students, it is required to schedule N_l professor-class pairs within a time limit of N_t time slots producing a legal schedule. A legal schedule needs to be found such that no professor, class, or student is in more than one place at a time, and no room is expected to accommodate more than one lesson at a time or more students than its capacity.

Because finding a feasible solution is NP-complete, we accept the possibility the algorithm will not find a feasible solution.

6.4.3 Operators

To come from a timetable to another one, operators are applied to change the timetable. The operators applied are:

Add	Shift a lesson from the dummy hour (not taught) to a week hour in such a way that the hard constraints are satisfied. The average number of students per lesson of this type should stay above the lower bound for this type.
Remove	This is the inverse operator of "Add". A lesson from a week hour is shifted to the dummy hour (not taught). The offered capacity should be sufficient.
Swap	Swap two lessons taught in the week.
Change resource	Change one of the resources (teacher or room) of a lesson.

The constraint for the "Add" operator that the average should be above the lower bound is forced implicitly by having such a total number of lessons (schedule and dummy hour together), that if all lessons are in the schedule, this constraint is still fulfilled. All operators are only applied if it does not result in an infeasible timetable.

6.5 Students matching level

If a timetable is known, individual students are assigned to the lessons that are offered. How this is is done is described in this section. Hard constraints for a matching are formulated, and an algorithm to get a feasible solution is given. To conclude, operators are formulated to slightly modify a current solution. Recall that the way of representing a matching was formulated in section 5.3.

6.5.1 Hard constraints for a matching

Hard constraints for a matching of students to a timetable are:

- All students can participate in a set of lesson such that their demands for lessons are met.
- Students are only matched to lessons that take place within their availability.
- There is no lesson that has more students matched to it that its capacity.
- If there is an order in which lessons should be attended, this order is respected.

6.5.2 Finding a feasible matching

To find a feasible matching, the timetable to which the students should be matched is temporary extended with a dummy day. This dummy day contains for every lessontype of the timetable x lessons, where x is the number of hours that is needed for that lessontype. All the lessons of the dummy day have an infinite capacity. Now the students are matched to lessons at the dummy day such that they participate in all the lessons required for them. Now, to find a feasible solution, we execute a simple local search:

Objective	We use a penalty function (so, to minimize). For every lesson that a student participates at the dummy day, we assign 1 penalty point.
Operators	We use 2 possible operator. One shifts a student participating in a lesson at the dummy day to a lesson of the same type at the timetable. The other shift a students participating in a lesson at the timetable to another lesson of the same type at the timetable. Off course, hard constraints are respected.
Stop	If the total penalty is 0, a feasible matching is found, and the local search will be stopped. The local search will also stop if the last u runs did not result in improvement.

If there is termination without a feasible solution, a feasible solution is not likely to exist. In further matching of students, overlap is therefore not taken as a hard constraint, but taken into account in qualifying the timetable (or directly request another timetable).

6.5.3 Operators

Swap students	This operator swaps the moment of participating in a lesson of a certain type for 2 students. If student s_1 participates in a lesson of type t at hour w_1, and student s_2 participates in a lesson of type t at w_2, student s_1 participates at moment w_2 and student s_2 participates at moment w_1 after applying this operator.
Move student	This operator moves a student participating in a lesson at hour w_1 to a lesson of the same type offered another time in the week.

Chapter 7

Implementation details

This chapter first gives some brief details on the implementation of the prototype. Secondly, the parameters used in during optimization are discussed. With this, the neighborhood space in the testworld is considered. This chapter concludes with the discussion of some results of the prototype.

7.1 Application details

A requirement for the project is to deliver a prototype. This prototype is created based on the process as formulated in section 2.4. This realized by implementing the fundamentals as formulated in section 4.1 and chapters 5 and 6. Input is read from .csv (comma separated values), and output written to .csv. Interaction takes place by means of a command shell. Instructions are printed, and user input can be entered on the command line.

The prototype is written in the object oriented programming language Java. Java is chosen because this is the standard at Educator (the Educator application is coded in Java). Another pro of choosing Java is the familiarity of the programmer with the general concepts of object oriented programming and the syntax of Java.

7.2 Simulated Annealing details

In simulated annealing, as formulated in section 6.1.2, a few settings have a major influence on the behavior. These are the parameters α, T and Q. In this section, we discuss the parameters as used in the application. Recall from chapter 6 that simulated annealing was applied at three levels. If we mention simulated annealing here, we say whether this is general for all the levels, or specific for a level.

7.2.1 Parameter T

The probability of accepting an impairment depends on the increase of the minimization function and on parameter T, since $p = exp\left(\frac{f(x)-f(y)}{T}\right)$. A rule of thumb is that at the start of the process, half of the disimprovements is accepted. However, the value for T that corresponds to a fifty percent probability of accepting such a disimprovement varies per problem and per instance of that problem. Hence, it is hard to decide a priori what the value of T should be. In the implementation, at all levels, we solved this by adding a phase prior to the actual optimization. During this phase, a good value for T is approximated. The weighting phase contains the following steps:

1. Estimate an initial value for T
2. for Q runs of simulated annealing
   - if $p < 0.5$, $T = \frac{T}{0.98}$
   - if $p > 0.5$, $T = T * 0.98$

An initial value for T can be estimated from the results of related instances.

7.2.2 Cooling

In section 6.1.2 the parameter T is modified (the cooling) by multiplying it by parameter α every Q runs. This way of cooling is called geometric cooling . We fix the value of α before the start of the application, for every simulated annealing level. The parameter Q is specified per level. A rule of thumb for the value of Q is 8 times the size of the neighborhood space. The size of the neighborhood space differs per level, and per problem instance of that level. Considerations concerning the neighborhood space, given below, are based on the testworld as defined in section 4.2, resumed in table 7.1.

Education periods	4
Sub periods (weeks)	10
Time units (hours) in sub period	5×10
Courses	22
Lessontypes per course on average	2
Required hours per type of lesson on average	2
Teachers	5
Rooms	8
Students	400

Table 7.1: Testworld

Course distribution level

At the distribution level, we formulated 3 operators (section 6.3.3):

Move	Move course to another education period.
Add	Offer an extra occurrence of a course.
Change resource	Change a resource (teacher, room) of a lesson (lesson is invisible, but considered to keep track of the utilization factor of resources).

Moving a course to another education period can reach a number of neighbors that is equal to the product of the number of course instances (where an instance is a course period couple) and the number of possible periods to move to. There are 4 education periods, so the number of possible periods to move to for a course is 3. Given that all the courses are assigned to at least one period, and at most to all periods, the number of course instances in the testworld varies between 22 and 88. Hence, the size of the neighborhood space for this operator is between 66 and 264. We will use 264. Offering a course in an extra period results in a neighborhood of 4 (number of periods) times 3 (periods that the course is possibly not offered) times 22 (number of courses) is 262. Because every course needs 2 (average number of lessontype) times 2 (average number of hours needed for a lessontype) times 5 (the times the entire program should be offered to be able to fully match the capacity with the demand of the students) is 20 lessonhours, every course needs 20 teachers and 20 rooms, which is independent of the number of periods the course is provided. The neighborhood of changing the teacher is 22 (number of courses) times 20 (number of teachers that can be changed) times 4 (number of teachers that could possibly replace the teacher) is 1760. Likewise, the neighborhood size of changing the room can be computed, which is 22 times 20 times 7 is 3080.

The sum of the operators is 5366, which is approximately the size of the neighborhood space.

Timetabling level

Operators at the timetabling level were defined (section 6.4.3):

Add	Shift a lesson from the dummy hour (not taught) to a week hour in such a way that the hard constraints are satisfied. The average number of students per lesson of this type should stay above the lower bound for this type.
Remove	This is the inverse operator of "Add". A lesson from a week hour is shifted to the dummy hour (not taught). The offered capacity should be sufficient.
Move	Move a lesson taught in the week to another moment in the week.
Change resource	Change one of the resources (teacher or room) of a lesson.

At timetable level, we have a number of lessons that are on the schedule and a number of lessons at the dummy hour. In a situation in which 5 courses are taught in an education period, 100 lessons (20 per course, as seen in section 7.2.2). The dummy hour

has approximately 25 lessons. Adding a lesson can be done in 25 (size of dummy hour) times 50 (hours in the week) is 1250 ways. The removing operator can be applied to any of the scheduled lessons, replacing this lesson to the dummy hour. This can be done in 100 ways, since there are around 100 lessons scheduled in the considered situation. Moving a lesson has 100 candidate lessons to be applied to (the scheduled lessons) and 49 possible of moving it to (all the hours except the origin of the lesson), so the neighborhood size here is 4900. Changing a resource can be split in changing a teacher and changing a room. Changing a teacher can be applied to all the scheduled lessons (100), by changing the current teacher to one of the 4 other teachers. Neighborhood size of changing a teacher is therefore 400. Likewise, the neighborhood size of changing a room is 700 (since there are 7 *other* rooms).

The total neighborhood size is (approximately) $1250 + 100 + 4900 + 400 + 700 = 7350$

Student matching level

Operators at the matching level were defined as follows:

Swap students	This operator swaps the moment of participating in a lesson of a certain type for 2 students. If student s_1 participates in a lesson of type t at hour w_1, and student s_2 participates in a lesson of type t at w_2, student s_1 participates at moment w_2 and student s_2 participates at moment w_1 after applying this operator.
Move student	This operator moves a student participating in a lesson at hour w_1 to a lesson of the same type offered another time in the week.

Every course has on average $\frac{3 \cdot 400}{5} = 240$ students (every student participates in 3 courses on average, there are offered 5 courses on average, and there are 400 students) and consists of 2 types on average. Those types take 10 hours each (2 hours with a student capacity of 240 is considered not possible). Size of the neighborhood space for the move operator is 5 (number of courses) times 2 (average number of lesson types per course) times 45 (number of possible pairs of lessons) times 48 (average number of students per lesson) times 2 (the direction of moving a student. The possible pairs of lessons is taken as the number of possible ways to take 2 lessons out of 10 ($\binom{10}{2} = 45$). A move is not allowed if it causes the student to consume both of his required lessons at the same moment. Since this is an approximation, we do not take that further into account. The neighborhood size of the move operator is: $5 \cdot 2 \cdot 45 \cdot 48 \cdot 2 = 43,200$. The swap operator much resembles the move operator. Only instead of selecting a direction and a student, a student pair needs te be selected. Since this can be done in $48 \cdot 48 = 2304$ ways, the neighborhood size for the swap operators becomes $5 \cdot 2 \cdot 45 \cdot 48 \cdot 48 = 1,036,800$.

Hence, the total neighborhood size at studentmatching level becomes: $1,036,800 + 43,200 = 1,080,000$

7.2.3 Stop condition

Two different stop conditions are defined. One of these gives whether the process is cold enough (so this cooling can be terminated). After this cooling process has been terminated, the solution is changed and heated again. The second stop condition defines when this reheating should be stopped.

The first stop condition is equal at all levels. If none of the last 10^3 disimprovements is accepted, the process is said to be cold enough. For the second stop condition, two different definitions are used. First one is simple: stop after a predefined number of reheatings. Second one is to stop if less than u of the last v restart resulted in an improvement. Parameters vary per experiment. Changing the solution between reheatings is done by applying a number of times a random operator. This number is referred to as "shufflesize". The chances for selecting an operator are distributed uniformly here.

7.3 Results

7.3.1 Course distribution level

Optimization of the distribution of courses among the education periods is done with the following parameters:

Q	45,000
α	0.97
T	1000
Stop condition	Less than 0.1% of disimprovements accepted
Process restarts	30
Shufflesize	20
Operator selection	Uniform distribution

Table 7.2: Distribution Parameters

Recall that this T value is not the actual start value of the optimization process, but the start value for the process of deciding what start value T should have (section 7.2.1). The result of a course distribution is given in Appendix D.1. In the table, for every period is given which courses should take place. To take the resource utilization into account, lessons are generated for the courses. The corresponding capacity is based on the underlying lessons of the courses.

7.3.2 Timetable level

Optimization of the timetable is done with the following parameters:

In Appendix D.2, an example of a timetable day is given. The lesson id given in the second column has the following syntax:

Q	60,000
α	0.97
T	1000
Stop condition	Less than 0.1% of disimprovements accepted
Process restarts	30
Shufflesize	300
Operator selection	Uniform distribution

Table 7.3: Timetabling parameters

[COURSE]_[LESSONTYPE NUMBER]_[SEQUENCENUMBER].

In the last column the expected number of students that will participate in this lesson is compared to the capacity of this lesson. This expectation is based on the matching of simulated students to this timetable.

Information concerning the timetable is printed on the pages in the appendix succeeding the timetable day. At the first information page is given how many simulations are done to predict the demand (pre simulations) and how many simulations of the students are done to match to the generated timetable (post simulations). 400 Students exist is the testworld, but it is possible that some of them are not interested in any of the scheduled courses. For example, if a student already completed all of the offered courses. The number of students that can not be supplied by any of the courses is given at the row "nOf unschedulable". Below this table is given per course the expected number of students (result of demand prediction), the number of subscriptions for this course, and the capacity that is used as a base for this timetable. This last parameter can differ from the result of the demand prediction if the person using the timetable engine has chosen to adapt the prediction. One table lower, some information on lesson level is given. Some values in this table have a succeeding value between brackets. The first value here expresses the the situation based on "expected", while the value in the brackets expresses the situation based on "planned for". Since no adaptation has been done to the prediction, the values are equal. The second column of the table gives the number of students that is expected for this lesson. Multiplied by the hours needed for this lesson (third column) we get the expected need of student places in a timetable. The net capacity gives the offered capacity, reduced with the student places that cannot be used. For example, if a lesson has to take place 2 times a week, for 50 students, 100 student places are required. Offering the lesson once a week with a capacity of 100 (or 2 of 50 at the same hour) will not do. This lesson gives a net capacity of 50, so another 50 student places need to be realized. UF is the abbreviation of "utilization factor" and expresses the used student places in relation to the available student places. The buffer gives how many extra places there are. The last table of this page expresses the weights of the goal functions. The weight functions are scaled such that using a weight of 1 for all of them as a start situation makes sense.

The next page gives some information about the quality of the timetable. In the first table, idle time for the simulated students that are matched is given. For this group of

students, the average idle time and the standard deviation is given. Furthermore, per amount of idle time is given how many of the matched students had that amount. The table below is build the same way, only here the deviation of the preferable day length for the whole week is expressed. For example, if a student has a preferable day length of 5 hours, one day of 3 hours and one day of 7 will give a deviation of 4, while 2 days of 5 will give no deviation (0). The last table is build the same way as the previous two and concerns the hour preferability of the students. This score is computed by taking hour reward of a student relative to maximum possible our reward for this student.

Information concerning the use of resources is given on the next page. For every resource type, a table is given (here: room and teacher). This table contains for every resource of that type the number of hours used in the timetable, the maximum number of hours that this resource could be used and the quotient of those.

7.3.3 Student matching level

Optimization of the timetable is done with the following parameters:

Q	8,000,000
α	0.97
T	1000
Stop condition	Less than 0.1% of disimprovements accepted
Process restarts	30
Shufflesize	10,000
Operator selection	Uniform distribution

Table 7.4: Student matching parameters

The result of the matching of students is a personal timetable for all of the (simulated) students. From these personal timetables, the information below the general timetable is computed. In Appendix D.3, one day of one of those personal timetables is printed. In the information below the timetable, "idle time" means the total idle time of this student in the schedule. Day length deviation expresses the deviation of the preferable day length, where this preferable length is in the brackets. The last row expresses how many preference points are scored (in the brackets the maximum possible preference points for this student).

It is possible for lessons to be coupled. This means that a student must participate in the required program together with a predefined number of fellow students. In the matching stage, students are matched in such a way that demands on those groupsizes are respected. The underlying groups of the matching are written to a text file. An example of this is shown on the second page in Appendix D.3. Here, lessontype 0 of course C_1021 is required two hours per week for participants. Those two hours are coupled, for groups of size 4-6. This means that it must be possible to compose the number of students that have selected the same couple of lessons of this type from

groups of 4-6. This can also be valuable information if there is a wish to couple the lessons beforehand.

Chapter 8

Conclusion

This project was driven by some questions of Educator, as defined in section 1.4. Along the way, all the questions are answered. We defined the planning process, containing the moments and subjects of interaction with the planning engine. An optimization model is defined, and used as a base for the planning engine. This optimization model is described in full detail in this thesis and answers the questions concerning optimization goals and quantification. Basic resources for a school are rooms and teachers. However, in the model this is defined in a generic way, so this can easily be extended. Scheduling of teachers is treaded as resource allocation, with some special configuration options. The prototype is validated by demonstrating it a few times at Windesheim (Educator customer). Educator is provided with enough knowledge to decide whether or not to build (since they decided to build based on this thesis).

Chapter 9

Continuation

Now that this project is completed, Educator is provided with insight of the problem, and a working prototype that they are willing to productize. The prototype works fine on a test data set, but is not tested intensively on a large real data set. First step will be to test it on real data and make it applicable for large data sets. If this is done, a final version must be implemented, that perfectly integrates with the other Educator components. In the end, Educator is extended with a modern planning engine that fits the needs of both classical education as on demand learning.

I will contribute to this process, since Educator offered me a job as "Operation Research Engineer", which I accepted. In this job, the above mentioned will be my primary concern.

Appendix A

Early scenarios

In order to gain a sharp view of the problem, we composed a number of different scenarios, each describing a way of tackling the problem. Of those scenarios, we pointed out both the positive and the negative sides. A comparison of the scenarios concludes this section. In this section, hard constraints and good timetables are mentioned. Hard constraints are for example the availability of teachers and rooms. Students having little idle time between lessons on the same day is probably a property of a good timetable. To define the hard constraints and what a good timetable is, a domain expert should be interviewed. During this interview, it is preferable to discuss a potentially good scenario. Such a scenario is composed prior to the sharp definition of the hard constraints and what a good timetable is. The early scenarios here are defined based on trivial hard constraints and trivial properties of a good timetable.

Scenario 1

The following steps define scenario 1:

Step 1 (crew scheduling)

After this step, each teacher has a list of lessons, ordered by preference to teach that lesson. On the list are only lessons the teacher is allowed to teach. This could cause a conflict, in the case there are more teachers having a particular lesson on the top of their list then the need for that lesson. To solve this problem, a priority is assigned to fulfilling the request of the teacher, for every teacher. This step could also be formulated the other way around, by assigning a list of preferences to each lessons, telling who is the preferred teacher to teach this lesson. Again, to handle conflicts, a priority is assigned to fulfilling the request of a lesson, for every lesson.

Step 2 (showcase timetable)

This step provides a showcase timetable, based on the preferences provided in step 1, the expected subscriptions and the hard constraints.

Step 3 (adapting timetable)

Making slight changes to the showcase timetable, based on the actual subscriptions and incidents (teacher has become ill for a long time).

Reflection scenario 1

Positive in this scenario is that teachers have the possibility to enter their preferences as well as the importance of them. The showcase timetable is very precise, which provides the students with certainty of which lessons they participate in and when that is going to happen.

The preciseness of the showcase timetable is also a negative side of this scenario. When providing such a precise timetable, you are committed to the promised times of offering, which reduces the size of the set of possible timetables strongly. The more the actual subscriptions differ from the expectation the showcase timetable is based on, the higher the risk the best possible timetable is no longer possible. Another negative side of this scenario is the imprecise way of asking time preferences of students. Students are only asked whether they want that lesson at that time or not, instead of asking when they want it.

A question that remains open: suppose there are two teachers, Isaac and Albert. Now Isaac is having a higher priority assigned to acknowledging his preferences than the Albert. If the situation occurs that a Isaac should teach a lesson one rank lower in his list of preferences or a Albert should teach a lesson 2 ranks lower in his list of preferences, what should be decided?

Scenario 1b

This scenario is like scenario 1, but the showcase timetable is more general. For example, only the day on which the lesson is given is provided. Assigning times is done after the students have subscribed.

An advantage compared to scenario 1 it that it preserves more freedom. The risk of throwing away nice timetables for no reason is significantly reduced. A drawback is that students do not know exactly when they can participate in which lesson.

Scenario 2

Step 1 (crew scheduling and student preferences)

The first step consists of two things:

- Crew scheduling. This is done the same way as in section A
- Students clear out their availability by means of a table containing the days and hours of the week. For every hour in the week a traffic light is shown:

- Green: Fine to have activities
- Yellow: Rather no activities
- Red: No activities at this hour

Green is the default color. If a student does not change the traffic light (which will occur), he is considered to be fine with all hours. An alternative for this traffic light is a $1 \ldots x$ scale. Advantage is the possibility for a student to give his priorities sharper. A drawback is the hardness of estimating this parameter for the students, if x is 10 for example. Another drawback is that preferences are more often not fulfilled, which does not encourage the students to share their preferences the next time. Besides this time preference, the student is able to fill in the courses he would like participate in, ordered by the students priority for that course. An alternative approach is to specify it more. After the student shared his preferences for the hours in the weeks, he can be provided with the option of putting in this preference for each lesson separately. For example he does not want to participate in a lecture of mathematics in the morning because he is not awake enough.

Step 2 (automatic subscription)

Based on the preferences of the students, the preferences of the teachers and the hard constraints a timetable is generated. Students are automatically subscribed based on their preferences.

Step 3 (adapting the timetable)

Adapting the timetable if rare events occur, like long illness of a teacher.

Reflection scenario 2

Very important to make this scenario function well is the participation of the students. To realize this, it should be crystal clear to the students that their input matters and they are the ones having the benefits of it. A good approach for this purpose might be sending an automatically generated motivation of the timetable to the students involved. Such a motivation should clear out the choices made, for example why this student its preference has been ignored (you and 4 fellow students want mathematics on Monday, but 15 students want mathematics on Tuesday).

A positive side of this scenario is the possibility teachers have to enter their preferences as well as the importance of those. Students also have the possibility to enter their preferences in time and courses to participate in. With this knowledge, you take decisions instead of guesses, which makes the timetable better.

A drawback in this scenario is that at the moment the students subscribe (by entering their preferences), they do not know the precise time and day per lesson.

Scenario 3

Step 1 (prior showcase timetable)

Generate showcase timetable based on expected subscriptions and hard constraints. Teachers are coupled to lessons based on availability. After this step, a timetable is provided without teachers. The reason that teachers are coupled at the background is to ensure that there is a feasible solution belonging to the timetable without teachers.

Step 2 (crew scheduling)

The main teacher schedules the teachers. The underlying crew scheduling from step 1 could be used as a suggestion, in which the main teacher could swap.

Step 3 (adapting the timetable)

Making slight changes to the showcase timetable, based on the actual subscriptions and incidents (teacher has become ill for a long time).

Reflection scenario 3

Scenario 3 resembles scenario 1. They differ in the crew scheduling part. Scenario 1 allows the possibility to enter preferences for teachers, and then generates a timetable (having teachers assigned to lessons) taking into account those preferences. Scenario 3 first generates a timetable, and then allows the main teacher to schedule the teachers manually.

A positive side of this scenario is the possibility teachers have to enter their preferences as well as the importance of them. The showcase timetable is very precise, which provides the students with certainty of when they participate in what lessons.

The preciseness in the showcase timetable is also a drawback, because it reduces the numbers of possible timetables significantly. There is a serious risk the best timetable has been thrown away already. It is also possible that the perfect crew scheduling in the eyes of the main teacher is no longer possible within the prior showcase timetable, while there is a timetable with this perfect crew scheduling. Even if this perfect timetable is possible within the prior showcase timetable, it might be a hard task for the main teacher to oversee the possibilities if the problem size increases.

Scenario 3b

This scenario is like scenario 3, but the showcase timetable is more general. For example, only the day on which the lesson is given is provided. Assigning times is done after the students have subscribed.

An advantage compared to scenario 3 is that it preserves more freedom. The risk of throwing away nice timetables for no reason is significantly reduced. A drawback is that students do not know exactly when they can participate in which lesson.

Early scenarios compared

Comparing the moment of subscribing

The main difference between scenario 2 and the scenarios 1 and 3 is the moment of subscribing of the students. In scenario 1 and 3 subscribing happens after the timetable is generated, while this subscription happens before generating the timetable in scenario 2. The advantage of the approach in scenario 2 is that the preferences of the students could be taken into account, and no freedom is uselessly spoiled.

Initially there is a huge set of possible timetables. If we want to find the best timetable, we remove the timetables that will never be the best. We remove for example the timetables in which a teacher is scheduled at a moment he is not available. For some timetables it is not possible to differentiate between them, because we do not have enough knowledge. In this case, both timetables should be kept until there is enough knowledge to make such a differentiation, instead of randomly picking one. In the latter case there is a serious risk of throwing away a good timetable.

In case we generate a precise showcase timetable based on expectations (without investigation of student preferences), we strongly reduce the amount of possible timetables, because we are required to respect the times in the timetable. Here we do not reduce by throwing away bad timetables, but by throwing away timetables of unknown quality (the time a lesson is provided) and timetables of very uncertain quality (which courses are provided). Later, if more information is available, those timetables might turn out to be very good. Unfortunately those tables are no longer possible (because we have to respect the times in the given table).

This does not happen in scenario 2, where time is taken into consideration, and the demand for courses will be predicted more reliable. Timetables are removed if there is information to justify the removal. In scenarios 1 and 3 we cannot qualify a timetable based on the time a lesson is provided, only on the courses that are provided. A bigger risk of harm, caused by divergences in subscription compared to expectation, occurs here.

Within scenario 1 and 3, the student can choose every period which course to subscribe to. The student could only enter his preferences binary: does he want mathematics Monday 11 am or not. In scenario 2 the student is able to enter preferences for when he wants mathematics.

An advantage of scenario 1 and 3 is that students know immediately which lessons they have and when. In scenario 2 this can only be determined after the subscription deadline.

Comparing scenario 1 and 3

Scenario 1 and 3 differ in the order of crew scheduling and publishing the showcase timetable. First generating a prior showcase timetable (scenario 3) has the advantage that it provides the main teacher with some ideas of how to do the crew scheduling. Disadvantage is that it takes away possible timetables. If the main teacher is not satisfied with his options for crew scheduling and he wants more options, the timetable should be created again. This situation will occur, for example, if the main teacher has the desire that teacher Isaac teacher mathematics and physics, but those lessons are taught at the same time. In scenario 1 the main teacher is not provided with ideas, but an easy to use form should compensate this inconvenience. The timetable is in this case closer to the preferences of the main teacher. Scenario 1 seems the better one, compared to scenario 3.

In both cases the *b* version (keeping the showcase timetable more general), will probably lead to better timetables. It will never lead to worse timetables, since the set of possible timetables at the original version is a subset of the set of possible timetables of the *b* version. The price paid for this is the precision of the showcase timetable. After all, this seems worth it.

Premature conclusion

The order of quality of the suggested scenarios is according to me: scenario 2, followed by 1(b), followed by 3(b). This is founded on the desire to generate a timetable as good as possible. Scenario 2 is in my opinion the scenario to build on.

Discussing scenario 2

To improve the scenario, I discussed it with Vincent Kok (Educator) and with Han Hoogeveen and Marjan van den Akker (University of Utrecht).

Vincent considered automatic subscription according to the preferences of the students not possible. A lot of students are lazy and will not enter their preferences in time. On the other hand, teachers and the administrator of the rooms need to know the timetable in an early stage. The impossibility is in the fact that the teachers and room administrator must be provided with the timetable earlier than the deadline for subscribing.

Discussing the scenario with Han and Marjan, the planning horizon was considered. The planning horizon in scenario 2 is the period the students subscribe to. This could get us into trouble. As an example, we discussed the situation at the University of Utrecht.

Students subscribe for a quarter of a year, and every year the curriculum repeats itself. A problem now occurs if you have a planning horizon of a quarter of a year, and you keep shifting a course to the next period. In the last period of the year, this becomes nasty, because the courses that have been shifted must be taught now. This will result in very bad timetables, or there are no feasible solutions left.

Han and Marjan considered the precise way students can give their preferences for times in the week a weak point, and proposed a longer interval. This way, the preferences of the students are more often acknowledged. The problem is also kept more tractable.

Conclusion

Scenario 2 seems the best scenario so far, but it has its shortcomings. In a next scenario, the problems that came up in the discussion of scenario 2 should be solved.

Appendix B

Visiting Windesheim

B.1 Visiting Windesheim I

To get a sharper view of the actual situation, I visited a domain expert (December 6th, 2007). This domain expert is Inge Strijker, ICTO-coordinator at the School of Health Care, which is part of Windesheim. Windesheim is an organization providing education, subdivided in ten schools. The schools vary in the competences that can be acquired.

To find out

During my visit to Windesheim, there are a few questions to be answered. First of all, the current way of working needs to be cleared out. Windesheim has started using Iris for timetabling in the spring of 2007 at all their schools, after using Untis. This switch did not turn out to be a clear success, regarding the fact that six out of the ten schools switched back to Untis at the end of 2007. Motivation for the switching back was the problems of Iris to generate a timetable in time. Some timetables were published at the day of the start of the period. The School of Health Care is one of the 4 schools still working with Iris. To support the generation of timetables, Educator tried to provide a service within the Educator software in which the involved people were able to put the constraints, for example that a sports lesson must take place in a sports room. This was used by Windesheim, but Iris regarded the data not useful and rejected it. Now a teacher fills in some excel file, which is used as input by Iris. It is useful to get clear why this went wrong, why the School of Health Care did not quit using Iris and why they preferred Iris to Untis in the first place (while everybody was used to Untis).

While describing the scenarios, constraints were mentioned like they were given, while they were still undefined. Hard constraints define the feasible timetables. Within the set of feasible timetables, the soft constraints define what a good timetable is. To find out what the constraints are, and if they are hard or soft constraints, a few questions need to be answered. Those questions are summed in Appendix B.

Feedback from Windesheim to the current idea of how to tackle the problem is desirable. The scenario presented is scenario 2.1, which is scenario 2, adapted based on section A.

Scenario 2.1

Step 1 (scheduling the year)

In the first step the courses are divided among the periods of the year, based on an expected set of students for each course. Expectations come from the profile of students. A student profile contains information that is valuable for predicting this students subscription, such as the courses he has already past, school he belongs to, etc.

Step 2 (crew scheduling and student preferences)

Investigation of teaching preferences is done. The way this is done is not adapted compared to scenario 2.

The other main thing in this step is investigating the preferences of the student. A student clears out his time preference by means of a table containing day parts of the week. A day part is for example morning, afternoon and evening. Then he clears out the course preferences by filling a list of courses, ordered to priority of how much he wants to participate in that course.

Step 3 (showcase timetable)

The showcase timetable is generated, based on the preferences of the students (which courses and preferred moment) and the preferences of the teachers (which lessons).

Step 4 (subscription)

The students receive a subscription proposal (fitting their preferences) with the timetable belonging to it. They can directly confirm this, or adapt and submit.

Step 5 (adapting the showcase timetable)

The showcase timetable is adapted. This is for example necessary if subscriptions differ from expectations. If allowed and necessary, some lessons do not take place. If necessary, a lesson could be offered an extra moment in the week.

At Windesheim

Educator and Iris

At the School of Health Care, Iris is used to generate timetables. As mentioned in section B.1, Educator BV its attempt to provide a possibility to put in constraints in Educator has failed. Discussing this, Inge Strijker explains that initially they were willing to enter

the constraints into the form of Educator. This was done by a temporary employee, using excel files of the teacher as a source. It turned out to be pretty much work. The result of the Educator forms could not be used by Iris directly. Information in the forms had to be transformed again, this time to the preferred format of Iris. Facing this, the decision was made to translate directly from the excel files to Iris, thereby not using Educator timetable constraints anymore.

Constraints asked by Educator were not satisfying. The expertise field for example is a text field, where a selection option for choosing the lessons the teacher is allowed to teach would be more suitable. Also the number of groups participating in a lesson is to select, where the minimum and maximum subscriptions for a group would be better (and then decide the number of groups to form). In the future, a timetable is purely individual. For now, the option of making timetables for groups should be present. That is not the case, which makes it unsuitable for the current situation.

Substituting Iris for Untis was based on the expectation that Untis could not handle "on demand" learning. Iris was also expected to generate better timetables. Six out of ten schools already switched back. Reason for this is the unacceptable lateness of the timetable publication. It has happened that timetables were published on the day the period started (the process of generating the best timetable took 52 hours). The School of Health Care still uses Iris. Timetables were published in time here, and students consider the timetables of Iris better then the ones Untis generated.

A good timetable

A possible parameter for the quality of timetables is the idle time between lessons on the same day students have. This should be minimal and is considered very important. Minimizing this idle time for teachers is not an issue. This is actually the other way around. Teachers do not like to teach in one piece. Their hours could be divided over the day, so the can do some preparations between their activities.

Using as little rooms as possible is only important above a certain threshold. At Windesheim, there are rooms available for every school. They can be used, and it is not an issue to use these rooms as little as possible as long as you do not use more than there is available. This constraint is not hard, because in case there are not enough rooms, rooms can be rented from other schools. This option costs money, and should be used as little as possible. The utilization factor of all the rooms of all the schools in Windesheim is 80%, but due to bad timetabling, there is a shortance of rooms at some moments during the day. There could be no different lessons in the same room at the same time. For exams, this is allowed.

Taking into account at what time the students prefer to have class is not done at the moment, but the response to the suggestion is very positive. If this is possible, this would be nice feature. The hours a teacher is hired to teach does not have to be obtained in an artificial way (for example by decreasing the students per lesson if a teacher has

to little lessons to teach). Class size is not important as long as this stays within the size boundaries.

There is significant travel time between the rooms. Travel time is currently not taken into account at qualifying the table. It would be desirable to do this. Apart from the breaks, there is no time between classes (the end time of a class equals the starting time of the next class). The common way of handling this, is a request of the student if he can leave the class up to ten minutes before the end if he has his next class in a room far away. A timetable with little travel time is desirable.

At the Windesheim School of Health Care the entire curriculum is offered in a semester (two curricula in a year). The planning cycle is therefore half a year. Every semester there is a possibility for students to start. The semester contains 2 education periods (a year is build of education period 1 to 4). At the start of the semester, students subscribe for both the periods. It is possible for a student to participate in courses at other schools, but he will do that for the full semester (a "minor"). This student can, for the timetable generation, be considered as a student of that particular school.

There are enough teachers at the School of Health Care. The demands to the sizes of a class depend only on the lesson, not on the person teaching it. A teacher has an availability and a number of hours he wants to teach. During the generation of a timetable, the teaching hours can be divided over the available time of the teacher. There is no preference of the teachers how to do this. This freedom is not used right now. The exact days and times when a teacher is going to act has to be filled in, while he is indifferent about the moment (as long as it is within his availability). Attempting to end up with a timetable as good as possible, the employees fill in a few random distributions of the hours to teach among the teacher his availability.

The situation can occur that a showcase timetable is published, but the lower bound to the number of subscriptions is not reached. If this lesson is provided multiple times on different moments in the week, is it allowed to merge those lessons? If this lesson is only provided once, is it allowed to cancel? There is no policy for this case at the School of Health Care. It has occurred that a minor was canceled due to a lack of subscriptions, but that was just decided for that situation.

The ten schools of Windesheim have something to do with each other when some school runs out of rooms. Concerning teachers, courses and students the schools are independent. It is probably the best to connect the ten timetable engines working for the schools to each other, to get an optimal distribution of rooms among the schools. In the current situation, the level of cooperation of the schools seems to low to achieve this.

The duration of lessons at the School of Health Care is standard at the moment, although this can differ with different courses. There is no need to put an upper bound

on the number of hours a lesson is provided on the same day. The moment a lesson is offered in a week is not important. Teaching the same lesson multiple times a day is no problem, and spreading the lessons over the week is not considered an advantage. Courses can have precedent relations, and sometimes it is preferable to explicitly bind a course to a period. Within this period, it is preferable to put in the weeks a lesson can be offered. Within the week, it must be possible to assign the day and hour the lesson is offered in a week.

The output

If the following information can be consulted, there is enough information:

- A student can consult his personal timetable
- A teacher can consult his personal timetable
- A timetable for each room can be consulted
- Overview per period/week/day/hour
- A timetable for predefined groups does not seem necessary (per student seems to do the job and in the future students probably do not have groups)

Reflecting scenario 2.1

The response to scenario 2.1 at Windesheim was quite positive. The need for the precise way the teacher can enter their preferences might be a little overkill, although the idea was considered useful. Current practice is that teachers are satisfied if they are not scheduled out of their availability.

Considering the situation, some second thoughts came up. In the proposed scenario the planning cycle is a year, while the students subscribe for quarters of the year. For this reason, scenario 2.1 starts with dividing the courses of the curriculum among the quarters of the year. At the Windesheim School of Health Care the curriculum is half a year, and the students also subscribe for half a year.

If subscription takes place for more than one education period at once, it might be a good idea to distribute the courses among the education periods after subscription has taken place. Big advantage of this approach is that the information gained by the student subscription can be used in deciding to which education period a course should be assigned. A drawback is the possibility of an unequal distribution of courses in a students individual timetable. The way courses are distributed is done in such a way that the best timetable is realized. This is also based on an equal distribution of the courses among the education periods for students. However, it can occur that a student sees his courses unequally divided among the education periods. This student probably has a rather unusual course selection, and suffers from being a minority.

There are two ways of solving this problem. First, we can make it possible for the student to select more courses then he can participate in, and assign the number of courses to follow to this student such that the course distribution is equal among the education period . Within his selection, the student is guaranteed to participate in a number of courses. The remainder of the courses he has to follow will be selected out of the remaining courses.

Another way of solving this problem, is distributing the courses among the education period prior to the student subscription, based on student profiles, availability of teachers and availability of rooms. If a course without precedent relations is provided multiple times, it is preferable to divide this among the education periods.

A problem with the first solution is that the mentor has to approve the PAP of the student. In the first solution it is not clear what courses the student will participate in. The teacher needs to approve all possible combinations, which is undoable. Another problem is that, even if you guarantee less courses to the student, the possibility of an unequal distribution is still there.

A drawback of the second solution is that there is the possibility that you make a wrong choice because you do not know the preferences of the students.

All taken into account, the second solution seems the best. Distribution of the courses among the planning cycle will be done based on student profiles.

B.2 Visiting Windesheim II

During my first visit to Windesheim, Inge Strijker told me that some of her answers were applicable only for the situation at the School of Health Care. Some general questions about Windesheim remained open. For this reason, she recommended to speak to someone familiar with the situation at the entire Windesheim. This person is Viola van Drogen, who is a member of the education information management. She is familiar with the planning situation at Windesheim in general. I visited Viola van Drogen at January 2nd, 2008.

To find out

This visit should provide a broader view of the problems with Iris. Second thing is investigating the generosity of the situation at the School of Health Care. Furthermore, I want to ask some subjective questions already asked to Mrs Strijker. At last, I want to request Mrs Van Drogen to provide me with the required data. The guideline with questions is in Appendix C.

Feedback from Windesheim for the current idea of how to tackle the problem is desirable. The idea to present at Windesheim is basically the same idea as presented at the first visit to Windesheim (section B.1), only this time I will be more general. For example, I will not consider a year, but a planning cycle instead.

At Windesheim

Iris

Spring 2007, Windesheim decided to provide on demand learning. At that time, Untis was used. For the start of on demand learning, a new investigation for the best timetabling software was done. Iris seems suitable for the job, and was selected as the new software package. After some problems, Iris was substituted for Untis by six out of ten Schools. Untis adapted their software to make it suitable for on demand learning. The big mesh up that made the Schools decide to drop Iris, was caused by the combination of introducing a new education system (on demand learning) and new software (Iris).

General Situation

At the School of Health Care, the curriculum takes half a year. This is not common for Windesheim. For example, the School of Information Sciences has a curriculum of a year. It is common for all the schools to subscribe for the entire semester, containing two education periods of equal length.

The situation concerning the rooms differs a little from the situation as understood after the first visit to Windesheim. There are rooms specifically for a school (reservation domain). Those rooms can be used as much as they want. Then there are general Windesheim rooms (timetable domain). Those are not owned by a school. Every period those rooms are divided among the schools. For those rooms the utilization factor is around 80% for all of the Windesheim schools together, but this is an avarage taken over the entire day (8:30 am - 5:30 pm). From 9:30 am - 1:30 pm, there is a shortance of rooms. This is a serious shortance, concerning the use of emergency rooms and some rooms, with initially other purposes, turned into classrooms. Schools do not have to pay extra for rooms rented from other schools. Schools pay for the rooms they use, unregarded whether this room was assigned to that school or not. The drawback of using rooms of other schools is that the school is dependent of the availability of rooms at other schools. That availability is low from 9:30 am - 1:30 pm. Even if the utilization factor for those hours for a specific school is not that high, schools switch hours to those hours (because those are considered nice). Switching to 9:30 am - 1:30 makes it hard for other schools to rent a room at that hour. This is bad, because the hour can be necessary instead of nice for another school (availability of teachers for example).

Rooms are not the only common thing between schools. There are teachers teaching at more then one school. However, timetabling happens without interaction. A teacher

is available some days at school A and some days at school B. For this, timetabling can be done autonomously for the schools. There are enough teachers at Windesheim.

The possible teaching hours are from 8:30 am - 5:30 pm on weekdays, for regular students. Part-time students get classes from 5:30 pm - 10:30 pm (weekdays) and on Saturdays.

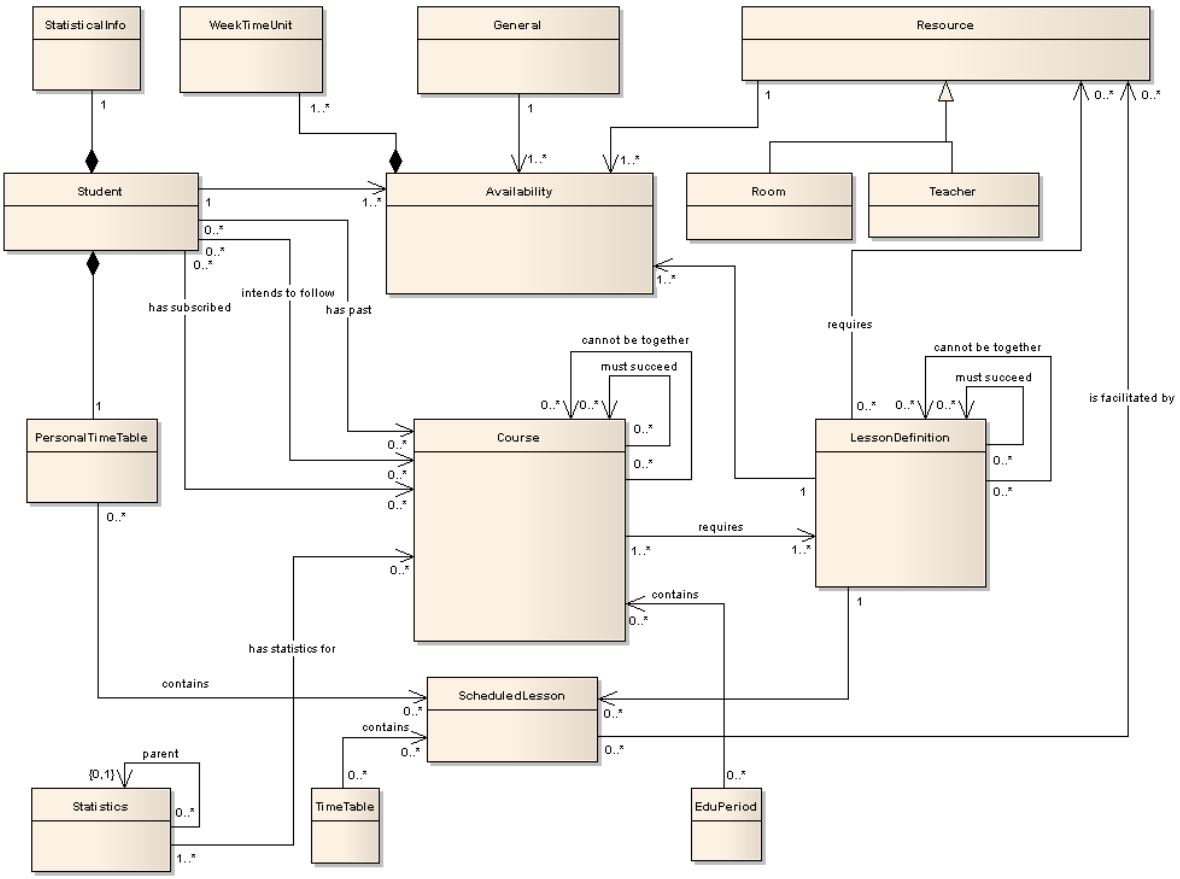
As is the case at the School of Health Care, there is no general policy for the case of not reaching the lower bound for participants of a lesson.

Exams are scheduled independent of the regular timetable. The last two weeks of an education period does not contain classes, only exams.

When verifying the current idea of what a good time table is, no new things came to light. Viola van Drogen also confirmed that there is enough information in the current format of the timetable output.

Appendix C

Data model



Appendix D

Results

D.1 Course distribution

Period 1

Course instance id	Course description	Capacity
C1005_0	Stotteren en stemstoornissen	165
C1004_0	Dysfagie	98
C1001_0	Afnemen en Interpreteren van testen	124
C1000_0	GGD	126
C1014_0	Neuropsychologie en -linguistiek	87
C1012_0	Nog beter horen	108
C1002_0	Anatomie en fysiologie	131

Period 2

Course instance id	Course description	Capacity
C1003_0	CVA-Ketenzorg	91
C1007_0	Articuleren en stemgeven	130
C1009_0	Niet meer kunnen wat kon	200
C1010_0	Voeren van tweegesprekken	222
C1013_0	Specifieke logopedische tools	107

Period 3

Course instance id	Course description	Capacity
C1019_0	Project	130
C1015_0	De Beroepsspreker	189
C1017_0	Communiceren over taal	142
C1016_0	Anamnese afnemen	127
C1006_0	Agenda van de Maatschap	203

Period 4

Course instance id	Course description	Capacity
C1020_0	Werkgroep WVAO	147
C1011_0	EBP en ICF Logopedie	150
C1008_0	Leerpsychologie	127
C1021_0	Werkgroep OIZ	139
C1018_0	Werkplekleren	121

D.2 Timetabling

Day:	**Monday**			
Hour	**Lesson**	**Teacher**	**Room**	**Exp.|Capacity**
8:30-9:30	C1001_1_S25a	Leonardo da Vinci	A034	30.0|90
8:30-9:30	C1000_2_S64	Alan Turing	E241	6.0|12
8:30-9:30	C1004_0_S56	Antoni van Leeuwenhoek	E227	10.0|10
9:30-10:30	C1001_1_26a	Leonardo da Vinci	A034	30.0|90
9:30-10:30	C1001_0_S56	Albert Einstein	E226	20.0|20
9:30-10:30	C1004_0_S58	Antoni van Leeuwenhoek	E322	16.0|16
9:30-10:30	C1000_2_B19	Alan Turing	E241	12.0|12
10:30-11:30	C1002_0_S74	Antoni van Leeuwenhoek	A076	63.0|90
10:30-11:30	C1004_1_S30	Alan Turing	A049	30.0|30
10:30-11:30	C1000_2_S102	Leonardo da Vinci	E322	16.0|16
11:30-12:30	C1004_0_S55	Albert Einstein	E322	16.0|16
11:30-12:30	C1000_1_S21	Antoni van Leeuwenhoek	A049	30.0|30
11:30-12:30	C1005_0_S16	Alan Turing	A034	90.0|90
11:30-12:30	C1000_2_S45	Leonardo da Vinci	E227	10.0|10
13:00-14:00	C1000_2_B12	Alan Turing	E322	16.0|16
13:00-14:00	C1000_1_B3	Antoni van Leeuwenhoek	A076	90.0|90
13:00-14:00	C1002_0_B11	Leonardo da Vinci	A049	30.0|30
13:00-14:00	C1001_0_S26	Albert Einstein	E230	16.0|16
14:00-15:00	C1000_0_B3	Leonardo da Vinci	A034	84.0|90
14:00-15:00	C1000_2_S69	Isaac Newton	E227	10.0|10
14:00-15:00	C1004_0_B20	Antoni van Leeuwenhoek	E322	16.0|16
14:00-15:00	C1000_2_S73	Alan Turing	E241	12.0|12
15:00-16:00	C1000_2_B29	Leonardo da Vinci	E322	16.0|16
15:00-16:00	C1004_1_S39	Alan Turing	A076	81.0|90
15:00-16:00	C1000_0_S19	Antoni van Leeuwenhoek	A034	31.0|90
15:00-16:00	C1001_0_S31	Isaac Newton	E226	20.0|20
16:00-17:00	C1000_2_S80	Alan Turing	E241	12.0|12
16:00-17:00	C1002_0_S17	Antoni van Leeuwenhoek	A076	90.0|90
16:00-17:00	C1001_0_B7	Albert Einstein	E230	16.0|16
16:00-17:00	C1001_0_S22	Isaac Newton	E226	20.0|20
17:00-18:00	C1000_2_S108	Isaac Newton	E322	15.0|16
17:00-18:00	C1005_0_S13	Alan Turing	A034	71.0|90
17:00-18:00	C1004_1_B3	Albert Einstein	A076	36.0|90
18:00-19:00	C1004_0_B2	Antoni van Leeuwenhoek	E227	10.0|10
18:00-19:00	C1002_0_S41	Isaac Newton	A076	45.0|90
18:00-19:00	C1000_2_S106	Alan Turing	E322	16.0|16

nOf pre simulations	127
nOf post simulations	1
nOf unschedulable	14
nOf scheduled	386

Course	Expected	Subscr.	Planned for
GGD	196	0	196
Afnemen en Interprete	186	0	186
Anatomie en fysiologie	210	0	210
Dysfagie	174	0	174
Stotteren en stemstoor	205	0	205

Lesson	Expected	Hours	Exp. Hours	Net cap.	UF	Avail. pl.	Aver. buffer
GGD, lestype 1	196(196)	1	196(196)	390	0.5 (0.5)	390	206
GGD, lestype 2	196(196)	2	392(392)	510	0.77 (0.77)	510	142
GGD, lestype 3	196(196)	3	588(588)	604	0.97 (0.97)	604	52
Afnemen en Interprete	186(186)	2	372(372)	420	0.89 (0.89)	420	50
Afnemen en Interprete	186(186)	2	372(372)	540	0.69 (0.69)	540	170
Anatomie en fysiologie	210(210)	4	840(840)	1080	0.78 (0.78)	1080	256
Dysfagie, lestype 1	174(174)	2	348(348)	354	0.98 (0.98)	354	2
Dysfagie, lestype 2	174(174)	3	522(522)	900	0.58 (0.58)	900	351
Stotteren en stemstoor	205(205)	2	410(410)	510	0.8 (0.8)	510	84

Type	Weight
Idle time	1
Overlap	1
Utilization factor	1
Distribution factor	1
HourPreferability	1
Course utilization factor	1

Idle time

Mean	0.25
Std	0.6

Hours idle time	nOf simulated students	Aver. students	Percentage
0	315	315	81.61
1	52	52	13.47
2	12	12	3.11
3	7	7	1.81

Day length (deviation of preferable daylength)

Mean	3.87
Std	2.39

dayLength deviation	nOf simulated students	Aver. students	Percentage
0	17	17	4.4
1	43	43	11.14
2	72	72	18.65
3	52	52	13.47
4	64	64	16.58
5	48	48	12.44
6	42	42	10.88
7	16	16	4.15
8	15	15	3.89
9	6	6	1.55
10	8	8	2.07
11	2	2	0.52
12	0	0	0
13	1	1	0.26

Hour preferability

Mean	88.50%
Std	7.25%

Hour preferability	nOf simulated students	Aver. students	Percentage
100.00%	51	51	13.21
90.0% - 100.0%	123	123	31.87
80.0% - 90.0%	170	170	44.04
70.0% - 80.0%	38	38	9.84
60.0% - 70.0%	4	4	1.04
50.0% - 60.0%	0	0	0
0.0% - 50.0%	0	0	0

Room	Scheduled hours	Available hours	Utilization factor
A035	3	50	0.06
A034	14	50	0.28
A076	24	50	0.48
A049	14	50	0.28
E227	21	50	0.42
E322	31	50	0.62
E241	21	50	0.42
E230	10	50	0.2
E226	13	50	0.26
A104	0	50	0

Teacher	Scheduled hours	Available hours	Utilization factor
IN	30	31	0.967741935
AE	31	32	0.96875
LV	29	30	0.966666667
AL	30	30	1
AT	31	33	0.939393939

D.3 Student matching

Day:	**Monday**			
Hour	**Lesson**	**Teacher**	**Room**	**Exp.|Capacity**
15:00-16:00	C1004_1_S39	Alan Turing	A076	81.0|90
16:00-17:00	C1002_0_S17	Antoni van Leeuwenhoek	A076	90.0|90
17:00-18:00	C1004_1_B3	Albert Einstein	A076	36.0|90
18:00-19:00	C1004_0_B2	Antoni van Leeuwenhoek	E227	10.0|10

TimeTable for Babet Dekker

Courses
Anatomie en fysiologie
Dysfagie

Idle time		0
DayLength dev. (pref.)	6.0(5)	
Time Pref Points (max)	75.0(90.0)	

Group	**C1021_0_S34, C1021_0_S22,**	
Id	Firstname	Lastname
S1248	Luna	Görtzen
S1351	Diego	Kramer
S1003	Evi	Berends
S1312	Tijn	Haverschmidt

Group	**C1021_0_S41, C1021_0_S34,**	
Id	Firstname	Lastname
S1071	Milan	Dijken
S1189	Pepijn	Kloppenburg
S1373	Lente	Bajnath
S1106	Owen	Kamp

Group	**C1021_0_S30, C1021_0_S44,**	
Id	Firstname	Lastname
S1290	Nova	Scholten
S1314	Evi	Brandenburg
S1159	Lise	Rietveld
S1149	Esther	Braeken

Group	**C1021_0_S30, C1021_0_S43,**	
Id	Firstname	Lastname
S1227	Simon	Gijssel
S1146	Nora	Leune
S1369	Aron	Spin
S1183	Maurits	Oordt

Group	**C1021_0_S44, C1021_0_S50,**	
Id	Firstname	Lastname
S1174	Alicia	Kolk
S1165	Mara	Schepers
S1397	Daniël	Postmus
S1340	Owen	Jong
S1081	Maurits	Pruyn
S1246	Jesper	Gessel
S1101	Martijn	Tunderman
S1024	Myrthe	Matel

Group	**C1021_0_S50, C1021_0_S22,**	
Id	Firstname	Lastname
S1371	Mare	Keijzer
S1063	Fabiënne	Creutzburg
S1280	Mandy	Put
S1117	Owen	Katerberg
S1209	Luna	Popken
S1383	Fabian	Esch

Bibliography

[BBKM06] Ruibin Bai, Edmund K. Burke, Graham Kendall, and Barry McCollum. A simulated annealing hyper-heuristic for university course timetabling. In *PATAT*, pages 345–350, 2006.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[dW85] D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2):151–162, February 1985.

[ECF97] M. A. Saleh Elmohamed, Paul D. Coddington, and Geoffrey Fox. A comparison of annealing techniques for academic course scheduling. In *PATAT*, pages 92–114, 1997.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[Lew07] R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30(1):167–190, 2007.

[MW66] N. Macon and E. E. Walker. A monte carlo algorithm for assigning students to classes. *Commun. ACM*, 9(5):339–340, 1966.