Technical University of Denmark

Comparing Solution Approaches for a Complete Model of High School Timetabling

Sørensen, Matias; Stidsen, Thomas Jacob Riis

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

Comparing Solution Approaches for a Complete Model of High School Timetabling

Report 5.2013

DTU Management Engineering

Matias Sørensen
Thomas Riis Stidsen

March 2013

DTU Management Engineering
Department of Management Engineering

Comparing Solution Approaches for a Complete Model of High School Timetabling

Matias Sørensen · Thomas R. Stidsen

Abstract A complex model of high school timetabling is presented, which originates from the problem-setting in the timetabling software of the online high school ERP-system Lectio. An Integer Programming formulation is described in detail, and a two-stage decomposition is suggested. It is proven that both of these formulations are $\mathcal{NP}$-hard. A heuristic based on Adaptive Large Neighborhood Search is also applied. Using 100 real-life datasets, comprehensive computational results are provided which show that the ALNS heuristic outperforms the IP approaches. The ALNS heuristic has been incorporated in Lectio, and is currently available to almost 200 different high schools in Denmark. Furthermore, a conversion of the datasets into the XHSTT format is described, and some datasets are made publicly available.

Keywords High School Timetabling · Modeling · Integer Programming · Decomposition · Adaptive Large Neighborhood Search

1 Introduction

The timetabling problem is perhaps the most important problem among the scheduling problems which high schools face. In this paper a complex model of the problem is presented, which originates from the problem-setting in the timetabling software of the online high school ERP-system Lectio. All specifications have been identified in close cooperation with high schools in Denmark. Thereby the developed model is tailored to the Danish case, but we expect it can easily be adapted to other variants as well. The model is 'complete' in the sense that it contains all relevant practical constraints, and there exists no difference among the model described in this paper and the one implemented in practice and made available to customers of Lectio.

Lectio is used by the majority of high schools in Denmark; Currently, 230 high schools are customers of Lectio, and 191 have bought access to the timetabling soft-

M. Sørensen (E-mail: msso@dtu.dk) · T.R. Stidsen
Section of Operations Research, Department of Management Engineering, Technical University of Denmark, Building 426, Produktionstorvet, DK-2800 Kgs. Lyngby, Denmark

M. Sørensen
MaCom A/S, Vesterbrogade 48, 1., 1620 Copenhagen V, Denmark

ware. This large customer base requires a model of the problem which is general enough to suit many different requirements, and which is also tractable by computer aided solution methods. This supports the recent trend of developing general models for timetabling problems (see Burke et al. (1998); Asratian and de Werra (2002); Özcan (2005); Causmaecker and Berghe (2010); Bonutti et al. (2010); Post et al. (2011, 2012a)). Furthermore, the timetabling problem of Danish high schools (denoted from now on as HSTTP) has not been formally described in the literature before.

The HSTTP concerns the construction of a feasible schedule which assigns lectures to timeslots and rooms, and maximizes individual preferences for students and teachers. The problem is usually handled by a single person (the *planner*) at each school. An efficient automated solution approach is important for the Danish high schools for the following reasons:

- Assisting the high school planners with decision support software will hopefully lead them towards better solutions and/or spend less time solving the problem.
- Maintaining employee satisfaction. Since a teacher usually teaches few timeslots each week, compared to the overall number of timeslots, it is sometimes possible to fulfill requests such as days-off, maximum number of teaching hours pr. day, etc. A timetable which fulfills such personal requests of each teacher will increase overall employee satisfaction. An opportunity to automate the solution approach allows the administration of the high school to see many diverse solutions, which will allow them to find solutions which fulfill such demands.
- A desirable timetable wrt. preferences of the students will lead to an overall better learning environment and also less drop-outs. Since each high school is paid by the government on the basis of number of students enrolled, a low number of student drop-outs is important.

The timetabling problem is considered at least annually by each high school, but some schools prefer to consider it several times each school year.

Three different solution approaches are proposed in this paper, of which two are based on a Mixed-Integer Programming (MIP) model and one is based on Adaptive Large Neighborhood Search (ALNS). The best performing solution approach is incorporated in Lectio and made available to all customers.

The paper is structured as follows: In Section 2 the HSTTP is described in detail, while simultaneously formulating a MIP model. Furthermore a simple decomposition approach is suggested, and both of these formulations are proven to be $\mathcal{NP}$-hard. A literature review of related problems and solution approaches is given in Section 2.1. An ALNS-heuristic is described in Section 3. Extensive computational results are presented in Section 4. A conversion scheme from HSTTP to the commonly used XHSTT format is described in Section A, which allows us to publish our problem instances.

2 The Timetabling Problem at Danish high schools

In the following the basic timetabling problem at Danish High Schools is described, which in Section 2.2 is formulated as a MIP model. In Section 2.2.2 the MIP is extended by several necessary side-constraints. A detailed formulation of a MIP model has the advantage that each constraint is described in a precise manner. Therefore we postpone the in-depth description of constraints to Section 2.2, and first give an overall description of the problem.

A high school has a number of teachers employed, and has access to rooms where teaching can be performed. Students are taught in classes of different subjects, and each class consists of a number of weekly lectures. Each day of the week is divided into *modules* where teaching is performed. A combination of a day and a module is denoted a *timeslot*. Students and teachers are preassigned to classes, as we assume the Teacher-Task Assignment problem (Lundberg-Jensen et al. (2008)) and the Student Sectioning problem (Kristiansen and Stidsen (2012)) have already been solved.

Lectures are represented by *events*, and each event must be assigned both a timeslot and a room. However it is also possible to combine several lectures for several classes into the same 'lecture'. This is used when classes should share the same room, e.g. in case of physical education, the sports venue. In such cases, a single event represents several lectures. Both the timeslot and the room of an event can be preassigned by the user of the system. An event has a set of eligible rooms which it can be assigned to. Often the set of eligible rooms equals the set of all rooms, but some events have special requirements, such as laboratories, sports venue, etc. See Figure 1 for an illustration of the notation used.

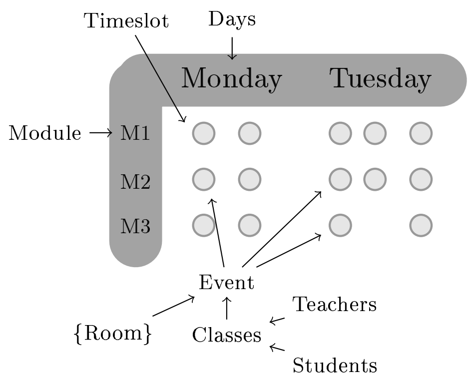


Fig. 1: *Notation used*

The timetabling problem essentially consists of creating a schedule for the entire school year, such that events are assigned a timeslot and an eligible room, and such that no clashes among students, teachers or rooms occur. Commonly the problem is solved by assuming that no difference exists among all weeks throughout the year, hence it is sufficient to plan only a single week. In this paper we also consider the case where the school decides to plan two consecutive weeks, which is elaborated in Section 2.2.4.

The Lectio interface allows the user to create chains of events, which forces some events to be in either the same timeslot or in contiguous timeslots. These will from now on be denoted as *EventChains*. EventChains are very flexible in the sense that they pose no restrictions on which events are chained together. Therefore they can be used to model a lot of special cases which the users of Lectio have requested. These include, but are not limited to, the following:

- A double-lecture can be set up by creating an EventChain consisting of two events for the same class which must be placed in contiguous timeslots. Similarly, triple-lectures can be created.
- Parallel double lectures for different classes.
- Grouping of elective classes in the same timeslot.

– Large projects where teaching of several classes are combined.

Another important aspect of this problem setting is its very practical nature. Due to feature requests and political decisions, the problem structure and specifications are occasionally changed. This gives rise to new constraints and changes in the objective weights. What is documented in this paper is how the problem currently looks, which has proven to be a quite stable formulation of the problem.

2.1 Related work

The definition of Class-Teacher Timetabling (CTT) is more than 50 years old, see Appleby et al. (1961) and Gotlieb (1962). In the original formulation, one is given a set of classes, a set of teachers and a set of periods. A class is defined as a set of students who follow the exact same curriculum. The goal is to find a schedule where classes meet teachers, fulfilling the teaching demand, subject to no teachers and classes being scheduled more than once in the same period. Furthermore, unavailabilities of teachers and classes in certain periods are given. In these periods, teaching is forbidden. This problem is pretty similar to the basic version of the HSTTP, except rooms are assigned to classes instead of teachers.

The survey Schmidt and Ströhlein (1980) covers early papers in the area of school timetabling. More recent surveys are Bardadym (1996); Carter and Laporte (1998); Schaerf (1999) and Pillay (2010). The conference series Practice and Theory in Automated Timetabling (PATAT) has contributed largely to the area (see Gendreau and Burke (2008); McCollum et al. (2010); Kjenstad et al. (2012)).

Lawrie (1969) was the first to formulate the basic problem as a column-based MIP. Since, integer programming has been used to model problems with more sophisticated requirements. Some recent contributions include Papoutsis et al. (2003); Avella and Vasil'Ev (2005); Avella et al. (2007); Birbas et al. (2009); Santos et al. (2010). However, integer programming is mainly used for timetabling due to its modeling strength, and not as an actual solution method. With the acknowledgment of modern IP solvers (see Bixby (2012)), this might be undergoing a change. The MIP formulated in this paper is among the most comprehensive models of school timetabling found in the literature.

The International Timetabling Competition 2011 (Post et al. (2012b)) considered a generalized version of High School Timetabling (based on the XHSTT format (Post et al. (2012a))). A contribution so far from this competition are several well-performing heuristics, see Fonseca et al. (2012), Kheiri et al. (2012) and Sørensen et al. (2012). Furthermore the competition proved the XHSTT format as a good foundation to build on for future research within the area. However, HSTTP contains constraints which cannot currently be modeled with XHSTT. In Section A the conversion from HSTTP to XHSTT is described in detail.

Lately, much research has gone into the university course timetabling problem, especially due to *The Second International Timetabling Competition 2007* (Gaspero et al. (2007); Lewis et al. (2007)). The most popular variant of the problem is the *Curriculum-based* Course Timetabling Problem (CCTP), where weekly lectures should be assigned to time periods and rooms. Recommended surveys regarding university course timetabling are Burke and Petrovic (2002) and Schaerf (1999). School timetabling and university timetabling are closely related (see (Carter and Laporte 1998, p. 5 (Table 1)), Nurmi and Kyngas (2008)). As a part of the HSTTP is to assign rooms to events, it seems that this problem is even more related than other school timetabling problems. The far most popular solution methods for the CCTP are heuristics (see Lewis (2008) for a survey). However, lately also integer programming has been tried. An interesting decomposition technique for the CCTP is described in Lach and Lübbecke (2008, 2012), where the assigning of rooms is postponed into a second optimization problem, while still maintaining optimality. This entails smaller IPs, which is proved to enhance solution times. Interesting approaches with integer programming for the university course timetabling problem are also found in Burke et al. (2008) and Burke et al. (2010). Daskalaki et al. (2004) presents a mathematical formulation using binary variables and several *operational rules* for a department at a Greek university, with convincing computational results.

Other papers which treats related problems: Dige et al. (1993) describes the timetabling problem at Danish primary schools, however this problem lack the complexity of the corresponding problem for high schools. Kingston (2010) covers the problem of assigning resources (e.g. teachers and rooms) after times has been assigned. In Prescott-Gagnon et al. (2009), Branch-and-Price as used as a repair method in a LNS framework for the VRPTW, with good results. A natural extension of the methods used in this paper would likewise be to combine the MIP approach and the ALNS heuristic.

The work presented in this paper has a practical character. I.e. the goal is to deploy the best possible solution approach to the customers of Lectio, and most of the described constraints originate more or less from feature-requests made by the users of Lectio. Papers which describe solution methods used in practice are not very common. Both Yoshikawa et al. (1996) and Kingston (2007) describe implementations which are used to create timetables for a few high schools. For universities, both Martin (2004) and Schimmelpfeng and Helber (2007) describe an IP-based approach tailored to a specific university.

2.2 Mixed-Integer Programming Formulation

A MIP formulation of the problem is given step-by-step, i.e. variables and parameters are introduced as needed. This formulation is stated such that a feasible solution always exists, as it is feasible to *not* assign an event to a timeslot and/or a room. However in such cases a penalty is given. Hence in principle, an optimal solution might be to not assign any events at all to timeslots or rooms, but in practice this is extremely unlikely. How we deal with events not assigned a timeslot is discussed under future research in Section 5. A feasible solution to the HSTTP is hence defined as a feasible solution to this MIP model.

The following sets are defined: Days $\mathcal{D}$, modules $\mathcal{M}$, timeslots $\mathcal{T}$, events $\mathcal{E}$, rooms $\mathcal{R}$ and classes $\mathcal{C}$. The set of entities is denoted $\mathcal{A}$, which includes both students and teachers. I.e. an entity $a \in \mathcal{A}$ is either a student or a teacher. Grouping these two types of entities in the same set allows us to write certain constraints in more compact form. To reduce problem size, students which are assigned exactly the same events are grouped into one super-entity. This is related to the concept of *curricula* in university course timetabling, and reduces the problem size considerably. Let $M_a \in \mathbb{N}$ denote the number of 'real' entities which entity a represents (so $M_a = 1$ if entity a is a teacher). In addition, let $\rho(i)$ denote the zero-based ordinal number of $i \in I$, e.g. if $I = \{a, b, c\}$, then $\rho(b) = 1$ with respect to set I.

Below constraints and objective function-terms of the problem are stated. First a basic model with well known constraints is introduced, and afterwards expanded to allow more specialized constraints. Finally various 'soft constraints' are added, which models different quality-metrics of the timetable. The notation used is lazy; $\forall e$ is short for $\forall e \in \mathcal{E}$, $\forall e \neq e'$ is short for $\forall e \in \{\mathcal{E} \setminus \{e'\}\}$, and $\sum_e$ is short for $\sum_{e \in \mathcal{E}}$. Parameters are written in uppercase (except for cost-parameters in the objective which are denoted with Greek letters), and variables are written in lowercase.

2.2.1 Basic model

The main decision variable is $x_{e,r,t} \in \{0,1\}$, which takes value 1 if event e takes place in room r in timeslot t, and 0 otherwise. Each event should be assigned one room and one timeslot, so we introduce the constraint

$$\sum_{r,t} x_{e,r,t} = 1 \qquad \forall e \tag{1}$$

To ensure a feasible solution exists, the set of timeslots and the set of rooms are extended by 'dummy' elements, which models that an event is *not* assigned a timeslot or a room, respectively.

$$\mathcal{T} = \{\mathcal{T} \cup \{t_D\}\}, \quad \mathcal{R} = \{\mathcal{R} \cup \{r_D\}\} \tag{2}$$

An assigning to either of these dummy-elements yields a big penalty (defined in Section 2.4).

The auxiliary variable $y_{e,t} \in \{0,1\}$ is introduced, which takes value 1 if event e is placed in timeslot t, and is constrained by the following

$$\sum_{r} x_{e,r,t} = y_{e,t} \qquad \forall e, t \tag{3}$$

This auxiliary variable is introduced for two reasons; 1) It simplifies the visual appearance of many of the constraints introduced further on. 2) It greatly reduces the number of non-zeros in the model, which reduces the memory-consumption when solving the MIP model.

Each entity can only participate in one event in each timeslot, except in the dummy-timeslot. Let $B_{e,a} \in \{0,1\}$ take value 1 if entity a is part of event e, and zero otherwise. The following constraint is imposed,

$$\sum_{e} B_{e,a} y_{e,t} \leq 1 \qquad \forall a, t \neq t_D \tag{4}$$

Each room (except for the dummy room) can only be assigned once to each timeslot (except for in the dummy timeslot). Furthermore, a room might be unavailable for teaching in certain time slots, for instance if it is shared by the high school and other institutions, or if its undergoing maintenance, etc. Let $G_{r,t} \in \{0,1\}$ take value 1 if room r is available at timeslot t, and zero otherwise. This constitutes the following constraint,

$$\sum_{e} x_{e,r,t} \leq G_{r,t} \qquad \forall r \neq r_D, t \neq t_D \tag{5}$$

An event might be locked to a specific timeslot or a specific room by the user. Let $LT_{e,t} \in \{0,1\}$ take value 1 if event e is locked to timeslot t, and let $LR_{e,r} \in \{0,1\}$ take value 1 if event e is locked to room r. The following constraints are imposed,

$$y_{e,t} = 1 \qquad \forall e, t, LT_{e,t} = 1 \tag{6}$$

$$\sum_t x_{e,r,t} = 1 \qquad \forall e, r, LR_{e,r} = 1 \tag{7}$$

Some events might require special rooms, i.e. chemistry lectures or physical education. Let $K_{e,r} \in \{0,1\}$ take value 1 if event e can be assigned to room r, and zero otherwise. The following constraint is imposed,

$$\sum_t x_{e,r,t} \leq K_{e,r} \qquad \forall e, r \tag{8}$$

It is not possible to assign a room to an event unless the event is also assigned a timeslot. This is because assigning timeslots is considered far more important than assigning rooms, and therefore it does not make sense to assign a room unless the event has a timeslot. Consider for instance an event assigned to the dummy timeslot. This event can be assigned to any room without violating the room conflict constraint (5). On the other hand, it is completely legal to assign an event to a timeslot, but not to a room. The following constraint is imposed,

$$\sum_{r \in \mathcal{R} \setminus \{r_D\}} x_{e,r,t_D} - \sum_r LR_{e,r} \leq 0 \qquad \forall e \tag{9}$$

This constraint specifies that if event e is not locked to any room, then it cannot be assigned to both a room different from the dummy-room r_D and the dummy-timeslot t_D. However if the event is locked to a room, it is legal to assign it to this room and the dummy-timeslot.

If an event is not assigned to a timeslot and/or a room, a penalty must be imposed. Denote this penalty by $\alpha_{e,r,t} \in \mathbb{R}^+$. The objective of the model therefore reads,

$$\min \sum_{e,r,t} \alpha_{e,r,t} x_{e,r,t} \tag{10}$$

2.2.2 Extended model

In this section, the model is extended to allow for constraints which arise from Event-Chains and other features of the Lectio interface.

In the Lectio interface, an event can be locked to more than one room, which clashes with constraint (1) in our formulation. This is handled in the following way: Assume event e_1 is locked to rooms r_1, r_2 and r_3. A 'super-room' entity $\overline{r}_1$ is created, which represents exactly these rooms. Hence event e_1 is locked to $\overline{r}_1$, and the set of rooms is extended accordingly $\mathcal{R} = \{\mathcal{R} \cup \{\overline{r}_1\}\}$. However this requires modification of the room conflict constraint (5), as an assigning to $\overline{r}_1$ forbids assigning to r_1, r_2 and r_3 for a given timeslot, and vice versa. Let U_r be the set of rooms which cannot be used simultaneously with room r, and let the set of rooms $\mathcal{R}$ be extended by all super-rooms. If room r is not a super-room, then it always applies that $r \in U_r$. I.e. for this example, $U_{\overline{r}_1} = \emptyset, U_{r_1} = \{r_1, \overline{r}_1\}, U_{r_2} = \{r_2, \overline{r}_1\}, U_{r_3} = \{r_3, \overline{r}_1\}$. Constraint (5) is modified to look as follows,

$$\sum_{e, r' \in U_r} x_{e,r',t} \leq G_{r,t} \qquad \forall r \neq r_D, t \neq t_D \tag{5'}$$

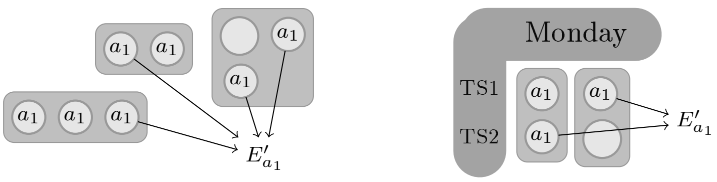
The EventChains are modeled by specifying that some events should be assigned the same timeslot as others, and that some events should be in contiguous timeslots. Let S_e be the set of events which must be assigned the same timeslot as event e, and let C_e be the set of events which must be assigned the timeslot following immediately after the timeslot assigned to event e. The following two constraint are added to the model,

$$y_{e,t} - y_{e',t} = 0 \qquad \forall e, e' \in S_e, t \tag{11}$$

$$y_{e,t} - y_{e',t'} = 0 \qquad \forall e, e' \in C_e, t, t', d_t = d_{t'}, \rho(t) + 1 = \rho(t') \tag{12}$$

where d_t denotes the day of timeslot t. The following example illustrates that constraint (12) is not completely sufficient for events which should be in contiguous timeslots; Assume event e_2 should follow in the timeslot immediately after event e_1, and that timeslot t_1 and t_2 is the first and second timeslot on some day, respectively. Constraint (12) specifies that if event e_1 is assigned timeslot t_1, then event e_2 must be assigned timeslot t_2. However we also need to constraint the problem such that it is forbidden to assign event e_1 to the dummy-timeslot, and e_2 assigned to timeslot t_1. This is done by extending the set of forbidden timeslots for an event, i.e. in this case forbid the assigning of event e_2 to timeslot t_1.

The EventChains imply further restrictions; Since the user might create EventChains which in itself causes a conflict between entities in terms of constraint (4), modifications to this constraint are needed. Let the set $E'_a \subseteq \mathcal{E}$ be the subset of events for which entity conflicts are checked for entity a. I.e. since we can a priori determine between which events in some EventChain an entity conflict will occur, we simply exclude some events from constraint (4). See Figure 2.



(a) *Three EventChains with several events where entity a_1 participates at same offset. Only a subset of events are checked for conflicts for entity a_1.*

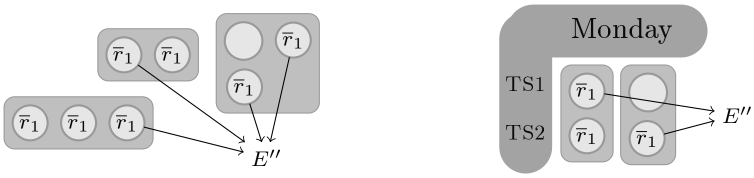
(b) *Two EventChains locked to timeslots, with several events where entity a_1 participates. Only a subset of events are checked for conflicts for entity a_1.*

Fig. 2: *Only some events are checked for entity conflicts*

The modified constraint is shown below,

$$\sum_{e \in E'_a} y_{e,t} \leq 1 \qquad \forall a, t \in \mathcal{T} \setminus \{t_D\} \tag{4'}$$

Furthermore, EventChains might also cause conflicts for a room in terms of constraint ($5'$), if several events are locked to the same room. See Figure 3. We introduce the set



(a) *Three EventChains with several events locked to room r_1 for same offset. Only a subset of these events are checked for conflicts.*

(b) *Two EventChains locked to timeslots, with several events locked to room r_1. Only a subset of these events are checked for conflicts.*

Fig. 3: *Room conflicts exceptions*

E'', denoting the events which should be checked for room conflicts. Constraint ($5'$) is modified to read

$$\sum_{e \in E'', r' \in U_r} x_{e,r',t} \leq G_{r,t} \qquad \forall r \neq r_D, t \neq t_D \tag{5''}$$

2.2.3 Timetable quality metrics

In this section, the model is extended by several quality metrics for a timetable. Most of these are modeled in the form of unwanted properties, whose (weighted) quantitative appearance should be minimized, commonly known as soft-constraints. However also a few hard-constraints are described, as some properties of a timetable are considered infeasible.

Idle timeslots An undesirable property of a timetable for an entity is *idle* timeslots. I.e. timeslots for an entity where no events are scheduled, but there is both an earlier and a later timeslot on that day where an event is scheduled, hence the entity must sit idle throughout some timeslots. By interviewing the high school planners, it is our experience that they especially consider idle timeslots for students to be undesirable. This might be due to (1) A student typically participates in so many events that a completely compact timetable seems to be possible, and/or (2) The high school planner believes that students are unlikely to do school-related tasks in an idle timeslot. For teachers, idle slots are also undesirable. However the high school planner will much prefer an idle timeslot for a teacher over an idle timeslot for a student.

Let the variable $h_{a,d} \in \mathbb{N}_0$ be the number of idle timeslots for entity a on day d. This variable is penalized in the objective as follows,

$$\sum_{a,d} M_a \beta_a h_{a,d} \tag{13}$$

where $\beta_a \in \mathbb{R}^+$ is the cost of an idle timeslot for entity a. Furthermore, let the variables $\underline{h}_{a,d} \in \mathbb{N}_0$ and $\overline{h}_{a,d} \in \mathbb{N}_0$ be the first and last timeslot where entity a is active on day

d, respectively. The following constrains are imposed,

$$\overline{h}_{a,d} - \underline{h}_{a,d} - \sum_{e \in E'_a, t \in \mathcal{T}_d} y_{e,t} + 1 = h_{a,d} \qquad \forall a, d \tag{14}$$

$$|\mathcal{M}| - (|\mathcal{M}| - \rho(t)) \sum_{e \in E'_a} y_{e,t} \geq \underline{h}_{a,d} \qquad \forall a, d, t \in \mathcal{T}_d \tag{15}$$

$$\rho(t) \sum_{e \in E'_a} y_{e,t} \leq \overline{h}_{a,d} \qquad \forall a, d, t \in \mathcal{T}_d \tag{16}$$

Equations (15) and (16) ensures that variables $\underline{h}_{a,d}$ and $\overline{h}_{a,d}$ are constrained properly. Notice that in case of entity a has no activities on day d (which naturally entails no idle timeslots), the value of $\underline{h}_{a,d}$ can be set to 1, to avoid $h_{a,d}$ to take value 1. Notice that these constraints use big-M notation, which is known to give bad LP-relaxations. This might have negative impact on solution times. A formulation without big-M notation is known, but it requires too many extra constraints to be applicable.

Unavailabilities For each event, it might be infeasible to assign it to certain times. This is used to prohibit teaching of certain classes at certain times. E.g. it is common that teaching of first year students is undesirable in the late modules on each day. Another example is to prohibit all teaching in the last module on Fridays, and only use this module in case a solution without it cannot be found.

Let $D_{e,t} \in \{0, 1\}$ take value 1 if it is feasible to assign event e to timeslot t. The following constraint for event unavailability is imposed,

$$\sum_{t, D_{e,t}=0} y_{e,t} = 0 \qquad \forall e \tag{17}$$

Furthermore, it is possible for the user to setup that certain timeslots are undesirable for a certain teacher. These 'soft-unavailabilities' are handled by simply adjusting the weight $\alpha_{e,r,t}$ for these timeslots for those events which the teacher is part of, see Section 2.4.

Days off It is quite common to require that all teachers have at least one day off. They can for instance use this day for preparation of future lectures. Let $F_a \in \mathbb{N}_0$ be the number of days off required for entity a (takes value 0 for all students). Notice that these days off are 'anonymous'. Let $f_{a,d} \in \{0, 1\}$ take value 1 if entity a has no events on day d, and zero otherwise. To make this variable take appropriate values, it is incorporated in constraint ($4'$). The rephrase of constraint ($4'$) is denoted ($4''$), which constraints the problem in equivalent way, but also makes $f_{a,d}$ take appropriate values,

$$\sum_{e \in E'_a} y_{e,t} + f_{a,d} \leq 1 \qquad \forall a, d, t \in \mathcal{T}_d \tag{4''}$$

The following constraint ensures entities are assigned to their required number of days off,

$$\sum_{d} f_{a,d} \geq F_a \qquad \forall a \tag{18}$$

Besides the required number of days off, it is generally preferred for teachers to have as many days off as possible. Therefore we also maximize the number of days off in the objective,

$$\sum_{a} M_a \gamma_a \left(|\mathcal{D}| - \sum_{d} f_{a,d} \right) \tag{19}$$

where $\gamma_a \in \mathbb{R}^+$ denotes the penalty for a day not being a day-off for entity a. Notice that the cardinality of $\mathcal{D}$ is incorporated to avoid this term to go below 0. Thereby the lower bound of the entire MIP is kept at 0.

The planners prefer that students have no days off. Therefore days off for students are penalized by adding the following term to the objective,

$$\sum_{a,d} M_a \delta_a f_{a,d} \tag{20}$$

where $\delta_a \in \mathbb{R}^+$ is the penalty for a entity a having a day off (takes value 0 for all teachers). Notice that since this expression minimizes $f_{a,d}$, constraint ($4''$) does not constrain $f_{a,d}$ sufficiently. Constraint ($4''$) only specifies that if an entity a has at least one event on day d, $f_{a,d}$ must take value 0. I.e. if an entity a has no events assigned on some day d, we need to make sure $f_{a,d}$ is forced to take value 1. This is done by the following constraint,

$$\sum_{e \in E'_a, t \in \mathcal{T}_d} y_{e,t} + f_{a,d} \geq 1 \qquad \forall a, d \tag{21}$$

Room stability A class would like all of its lectures to take place in the same room. We therefore aim at minimizing the number of different rooms assigned to events where a given class participates. Recall that if a lecture is locked to multiple rooms, these are grouped into one super-room, and let $v_{c,r} \in \{0,1\}$ take value 1 if class c is assigned to room r at least once and if room r is not a super-room, and zero otherwise. Let $Q_r \in \{0,1\}$ take value 1 if room r is a super-room, and zero otherwise. Let $J_{e,c} \in \{0,1\}$ take value 1 if class c is part of event e, and zero otherwise. The following constraint is imposed,

$$\sum_{e, t \neq t_D} J_{e,c} x_{e,r,t} - \sum_{e} J_{e,c} v_{c,r'} \leq 0 \qquad \forall r, r' \neq r_D, r \in U_{r'}, Q_{r'} = 0, c \tag{22}$$

This constraint specifies that if some event e is assigned room r and class c is part of event e, and room r cannot be used simultaneously with some room r', and room r' is not a super-room, then force variable $v_{c,r'}$ to value 1. Notice that usually $r = r'$. Let $s_c \in \mathbb{N}_0$ be the number of rooms assigned to class c minus one, i.e. the number of 'excess' rooms,

$$\sum_{r} v_{c,r} - 1 \leq s_c \qquad \forall c \tag{23}$$

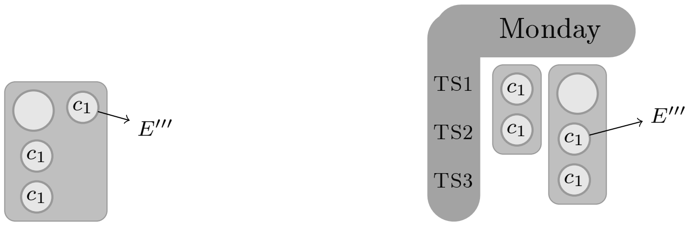
This following term is added to the objective,

$$\epsilon \sum_{c} s_c \tag{24}$$

where $\epsilon \in \mathbb{R}^+$ is the cost of each excess room assigned to a class.

Day-conflicts Each class can only be taught once each day, unless several events containing the same class are part of the same EventChain (e.g. double-lectures), or unless several events of this class are locked to timeslots on this day. Let $b_{c,t} \in \{0,1\}$ take value 1 if class c is part of at least one event on day d, and let $E''' \subseteq \mathcal{E}$ be the set of events for which day-conflicts are checked. All events are included in E''', with the following exceptions (see also Figure 4):

- Some events of the same class are part of the same EventChain. All of these, except one, is excluded from E'''.
- If multiple events are locked to timeslots on the same day, all of these events, except one, are excluded from E'''.



(a) *One EventChain where class c_1 participates in several events. Only one of these events is added to E'''*

(b) *Two EventChains locked to the same day. Only one event which c_1 is part of is checked for day-conflicts.*

Fig. 4: *Day conflicts exceptions*

The following constraint is added to the model,

$$\sum_{e \in E'''} J_{e,c} y_{e,t} \leq b_{c,t} \qquad \forall c, t \tag{25}$$

Day-conflicts of classes are thereby avoided by adding following constraint,

$$\sum_{t \in \mathcal{T}_d} b_{c,t} \leq 1 \qquad \forall c, d \tag{26}$$

Neighbor days for classes Another undesirable property for a timetable is neighbor-day-clashes for classes. Both students and teachers prefer that lectures of a class are spread throughout the week, for instance to allow more time for homework between lectures. Let $P_{d,d'}$ take the value 1 if day d and day d' are neighbor days, and zero otherwise. Neighbor-day pairs are Monday-Tuesday, Tuesday-Wednesday, etc., excluding Tuesday-Monday, Wednesday-Tuesday, etc. Let the variable $n_{c,d} \in \{0,1\}$ take value 1 if class c has a neighbor-day-conflict on day d, and zero otherwise. The following constraints are imposed,

$$\sum_{t \in \mathcal{T}_d} b_{c,t} + \sum_{t \in \mathcal{T}_{d'}} b_{c,t} - n_{c,d} \leq 1 \qquad \forall c, d, d', P_{d,d'} = 1, R_{c,d} + R_{c,d'} \leq 1 \tag{27}$$

If class c is locked to at least one event on two contiguous days, this is not defined as a conflict. Let $R_{c,d} \in \{0,1\}$ take value 1 if class c is locked to some event on day d, and zero otherwise. Neighbor-day conflicts are penalized by the following term in the objective (where $\zeta \in \mathbb{R}^+$),

$$\zeta \sum_{c,d} n_{c,d} \tag{28}$$

In case a class has few lectures, neighbor-day conflicts might even be infeasible. Let $N_c \in \mathbb{N}_0$ be the number of allowed neighborday-conflicts for class c, defined as follows:

$$N_c = \begin{cases} 0 & NC_c \leq 2 \\ w & NC_c = 3 \\ 3w & NC_c = 4 \\ 4w & NC_c \geq 5 \end{cases} \tag{29}$$

where NC_c is the number of EventChains where class c participates, and $w \in \{1, 2\}$ is the number of weeks being planned. The following constraint is added to the model,

$$\sum_{d} n_{c,d} \leq N_c \qquad \forall c \tag{30}$$

Teacher daily workload It can be preferred for teachers that they do not have to teach in all modules on a day. Let $W_a \in \mathbb{N}_0$ be the maximum number of lectures on a day for entity a (takes value 0 for all student entities). The following constraint is imposed,

$$\sum_{e, t \in \mathcal{T}_d} y_{e,t} \leq W_a \qquad \forall a, d \tag{31}$$

On the other hand, teachers do not like days with too few lectures. It is generally believed among the planners that a teacher should have at least two lectures on 'active' days, i.e. days with only one lecture are undesirable. In the following the model is constrained so days with only one active timeslot for an entity is penalized. Let $o_{a,d} \in \{0, 1\}$ take value 1 if entity a is a teacher and has only one lecture on day d, and zero otherwise. The following constraint is imposed,

$$2 - \sum_{e \in E'_a, t \in \mathcal{T}_d} y_{e,t} - 2f_{a,d} \leq o_{a,d} \qquad \forall a, d \tag{32}$$

The following expression is added to the objective,

$$\sum_{a,d} \eta_a M_a o_{a,d} \tag{33}$$

Deviation from previous solution When the users are running the algorithm, they prefer that the found solution does not deviate too much from the previous solution. A small penalty is imposed on assignments of timeslots and rooms which deviate from those of the previous solution. Let $t_P(e)$ and $r_P(e)$ be the previous timeslot and previous room assigned to event e, respectively. The variable $u_e \in \{0, 1\}$ take value 1 if event e was not assigned its previous timeslot, and zero otherwise. $p_e \in \{0, 1\}$ takes value 1 if event e was not assigned its previous room, and zero otherwise. The following constraints are imposed,

$$\sum_{t \in \mathcal{T} \setminus \{t_D, t_P(e)\}} y_{e,t} = u_e \qquad \forall e \tag{34}$$

$$\sum_{r \in \mathcal{R} \setminus \{r_D, r_P(e)\}, t} x_{e,r,t} = p_e \qquad \forall e \tag{35}$$

These variables are punished in the objective by the following,

$$\theta \sum_e (u_e + p_e) \tag{36}$$

If no previous solution exists, these constraints are omitted.

2.2.4 Two week schedule metrics

Planning two weeks instead of one gives twice the amount of timeslots, and thereby larger flexibility, which is prefered by some planners at the high schools. E.g. suppose a class in average should have five lectures each week. Instead of assigning five events to each week, four events could be assigned to the first week, and six events could be assigned to the second week. The planning of two weeks yields additional quality metrics.

Days off stability for teachers It is preferred to have the required days off for entities distributed equivalently among the two weeks, e.g. in case of 3 required days off, each week must contain at least 1 day off. This is done by the following constraint,

$$\Big| \sum_{d \in d(\underline{\mathcal{T}})} f_{a,d} - \sum_{d \in d(\overline{\mathcal{T}})} f_{a,d} \Big| \leq 1 \qquad \forall a \tag{37}$$

where $d(\underline{\mathcal{T}})$ and $d(\overline{\mathcal{T}})$ denotes days of the first and second week, respectively. This constraint is easily transfered into linear-form using two set of constraints.

Stability for lectures of classes Likewise, the distribution of lectures for classes should also be evenly distributed between weeks. Let $w_c \in \mathbb{N}_0$ be the number of events out of week-balance for class c. This is punished in the objective by

$$\iota \sum_c w_c \tag{38}$$

and is constrained by the following,

$$\Big| \sum_{e,t \in \underline{\mathcal{T}}} J_{e,c} y_{e,t} - \sum_{e,t \in \overline{\mathcal{T}}} J_{e,c} y_{e,t} \Big| - 1 = w_c \qquad \forall c \tag{39}$$

The complete model is shown in Model (40). Notice that all variables except for $x_{e,r,t}$, $y_{e,t}$ and $v_{c,r}$ can be stated as LP variables, as they will naturally take integer values. It is expected that not having integer requirements on these variables will facilitate a more efficient solution procedure.

HSTTP Mixed Integer Linear Program (40)

$$\begin{aligned}\min\quad & \sum_{e,r,t}\alpha_{e,r,t}x_{e,r,t}+\sum_{a,d}M_a\beta_a h_{a,d}+\epsilon\sum_c s_c+\zeta\sum_{c,d}n_{c,d}+\sum_{a,d}\eta_a M_a o_{a,d}+\\ & \sum_a M_a\gamma_a\left(|\mathcal{D}|-\sum_d f_{a,d}\right)+\sum_{a,d}M_a\delta_a f_{a,d}+\theta\sum_e(u_e+p_e)+\iota\sum_c w_c\end{aligned} \tag{40a}$$

s.t.

$$\begin{array}{llll}
\text{(time/room)} & \sum_{r,t} x_{e,r,t} & =1 & \forall e \qquad (40b)\\
\text{(aux. link)} & \sum_{r} x_{e,r,t} & =y_{e,t} & \forall e,t \qquad (40c)\\
\text{(entity conf.)} & \sum_{e\in E'_a} y_{e,t}+f_{a,d} & \le 1 & \forall a,d,t\in\mathcal{T}_d \qquad (40d)\\
\text{(room conf.)} & \sum_{e\in E'',r'\in U_r} x_{e,r',t} & \le G_{r,t} & \forall r\neq r_D, t\neq t_D \qquad (40e)\\
\text{(locked time)} & y_{e,t} & =1 & \forall e,t, LT_{e,t}=1 \qquad (40f)\\
\text{(locked room)} & \sum_{t} x_{e,r,t} & =1 & \forall e,r, LR_{e,r}=1 \qquad (40g)\\
\text{(feas. rooms)} & \sum_{t} x_{e,r,t} & \le K_{e,r} & \forall e,r \qquad (40h)\\
\text{(not only room)} & \sum_{r\in\mathcal{R}\setminus\{r_D\}} x_{e,r,t_D}-\sum_r LR_{e,r} & \le 0 & \forall e \qquad (40i)\\
\text{(same time)} & y_{e,t}-y_{e',t} & =0 & \forall e,e'\in S_e,t \qquad (40j)\\
\text{(cont. times)} & y_{e,t}-y_{e',t'} & =0 & \forall e,e'\in C_e,t,t',d_t=d_{t'}, \\ & & & \rho(t)+1=\rho(t') \qquad (40k)\\
\text{(n.d. conf.)} & \sum_{t\in\mathcal{T}_d} b_{c,t}+\sum_{t\in\mathcal{T}_{d'}} b_{c,t}-n_{c,d} & \le 1 & \forall c,d,d',P_{d,d'}=1, \\ & & & R_{c,d}+R_{c,d'}\le 1 \qquad (40l)\\
\text{(n.d. conf.)} & \sum_{d} n_{c,d} & \le N_c & \forall c \qquad (40m)\\
\text{(forbid. times.)} & \sum_{t,D_{e,t}=0} y_{e,t} & =0 & \forall e \qquad (40n)\\
\text{(idle slots)} & |\mathcal{M}|-(|\mathcal{M}|-\rho(t))\sum_{e\in E'_a} y_{e,t} & \ge \underline{h}_{a,d} & \forall a,d,t\in\mathcal{T}_d \qquad (40o)\\
\text{(idle slots)} & \rho(t)\sum_{e\in E'_a} y_{e,t} & \le \overline{h}_{a,d} & \forall a,d,t\in\mathcal{T}_d \qquad (40p)\\
\text{(idle slots)} & \overline{h}_{a,d}-\underline{h}_{a,d}-\sum_{e\in E'_a,t\in\mathcal{T}_d} y_{e,t}+1 & =h_{a,d} & \forall a,d \qquad (40q)\\
\text{(days off)} & \sum_{e\in E'_a,t\in\mathcal{T}_d} y_{e,t}+f_{a,d} & \ge 1 & \forall a,d \qquad (40r)\\
\text{(days off)} & \sum_{d} f_{a,d} & \ge F_a & \forall a \qquad (40s)\\
\text{(room stabl.)} & \sum_{e,t\neq t_D} J_{e,c}x_{e,r,t}-\sum_e J_{e,c}v_{c,r'} & \le 0 & \forall r,r'\neq r_D, r\in U_{r'},Q_{r'}=0,c \qquad (40t)\\
\text{(room stabl.)} & \sum_{r} v_{c,r}-1 & \le s_c & \forall c \qquad (40u)\\
\text{(day conf.)} & \sum_{e\in E'''} J_{e,c}y_{e,t} & \le b_{c,t} & \forall c,t \qquad (40v)\\
\text{(day conf.)} & \sum_{t\in\mathcal{T}_d} b_{c,t} & \le 1 & \forall c,d \qquad (40w)\\
\text{(worklimit)} & \sum_{e,t\in\mathcal{T}_d} y_{e,t} & \le W_a & \forall a,d \qquad (40x)\\
\text{(one lecture)} & 2-\sum_{e\in E'_a,t\in\mathcal{T}_d} y_{e,t}-2f_{a,d} & \le o_{a,d} & \forall a,d \qquad (40y)\\
\text{(prev. time)} & \sum_{t\in\mathcal{T}\setminus\{t_D,t_P(e)\}} y_{e,t} & =u_e & \forall e \qquad (40z)\\
\text{(prev. room)} & \sum_{r\in\mathcal{R}\setminus\{r_D,r_P(e)\},t} x_{e,r,t} & =p_e & \forall e \qquad (40aa)\\
\text{(class stabl.)} & \left|\sum_{e,t\in\underline{\mathcal{T}}} J_{e,c}y_{e,t}-\sum_{e,t\in\overline{\mathcal{T}}} J_{e,c}y_{e,t}\right|-1 & =w_c & \forall c \qquad (40ab)\\
\text{(d.o. stabl.)} & \left|\sum_{d\in d(\underline{\mathcal{T}})} f_{a,d}-\sum_{d\in d(\overline{\mathcal{T}})} f_{a,d}\right| & \le 1 & \forall a \qquad (40ac)\\
 & x_{e,r,t},y_{e,t},v_{c,r} & \in\{0,1\} & \qquad (40ad)\\
 & f_{a,d},b_{c,t},n_{c,d},o_{a,d},u_e,p_e & \in[0,1] & \qquad (40ae)\\
 & h_{a,d},\underline{h}_{a,d},\overline{h}_{a,d},s_c,w_c & \in\mathbb{R}^+ & \qquad (40af)
\end{array}$$

2.3 Two-stage formulation

Inspired by the approach taking in Lach and Lübbecke (2012), we propose to solve model (40) in two stages. I.e. in stage one, assign events to timeslots, and in stage two, assign events to rooms given the assigned timeslots. By this approach, the explosion in the number of variables caused by $x_{e,r,t}$ is avoided, as each stage can instead be modeled by a binary variable with two indices.

2.3.1 Stage One

In stage one, set $\mathcal{R} = \{\mathcal{R}_L \cup r_D\}$, where $\mathcal{R}_L$ is the set of rooms which are locked to at least one event. If an event e is not locked to a room, set the dummy-room as the only feasible room for this event, i.e. $K_{e,r_D} = 1$ and $K_{e,r} = 0 \ \forall r \neq r_D$. This forces all events which are not locked to a room to be assigned to the dummy-room. By this setting of parameters, exactly one feasible room exists for each event, which significantly reduces the number of variables in terms of $x_{e,r,t}$. I.e. $x_{e,r,t}$ is substituted by $K_{e,r} y_{e,t}$.

As we would like to not only generate good solutions by this approach, but also to generate lower bounds, it is assumed that each event can be assigned the best room possible. I.e. set

$$\alpha_{e,r,t} = \min_{r'} \alpha_{e,r',t} \tag{41}$$

Furthermore, the room stability constraints (40t) and (40u) are removed. These constraints are not redundant as they still applies to all locked rooms, but the constraints must be removed to generate a valid lower bound. I.e. some of the penalty produced by these constraints due to locked rooms might disappear when additional rooms are assigned in the stage two model.

Constraint (40aa) is redundant with this setting of parameters, so it is removed from the model.

With these modifications, Model (40) is solved to obtain a solution $y^*_{e,t}$ where events are assigned timeslots. The lower bound obtained by solving this model is a lower bound on Model (40). Notice that no constraints are imposed to ensure events can be assigned an eligible room in the next stage. This might give us worse solutions, but it is expected that the natural spread among the timeslots events are assigned to will also ensure a fair amount of rooms can be assigned without causing conflicts. It is expected that not adding additional constraints will give an easier model to solve.

2.3.2 Stage Two

In stage two, Model (40) is solved with the variables $y_{e,t}$ fixed as set by $y^*_{e,t}$. This turns the problem into a matter of assigning rooms to events, and this problem has a lot less variables than the original problem. All constraints are redundant, except for (40e), (40t), (40u) and (40aa). A feasible solution for this model is clearly a feasible solution to Model (40).

2.4 Weights

Below are listed the values of the weights used in the objective. These values have been selected on the basis of user inputs, and changes might come in the future. Let m_t denote the module of timeslot t.

$$\alpha_{e,r,t} = \begin{cases} 0 & LT_{e,t} = 1 \\ 60 & t = t_D \\ 2\rho(m_t) + 4\sum_a M_a B_{e,a} V_{a,t} & \text{else} \end{cases} + \begin{cases} 0 & LR_{e,r} = 1 \\ 10 & r = r_D \\ 2(p_{e,r} - 1) & \text{else} \end{cases} \tag{42}$$

where $V_{a,t}$ takes value 1 if it is undesirable for entity a to be assigned timeslot t (takes value 0 for all student entities), and $p_{e,r} \in \{1, 2, 3\}$ is the priority of room r for event e. Notice that the value of $\alpha_{e,r,t}$ is dependent on the module number of the timeslot, as early timeslots are generally more preferred than late timeslots. Furthermore if an event is locked to a timeslot or a room, no penalty for either of these assignings is given.

Remaining weights are defined as follows:

$$\beta_a = \begin{cases} 6 & a \text{ is teacher} \\ 7 & a \text{ is student} \end{cases} \tag{43}$$

$$\gamma_a = \begin{cases} 1 & a \text{ is teacher} \\ 0 & a \text{ is student} \end{cases} \tag{44}$$

$$\delta_a = \begin{cases} 0 & a \text{ is teacher} \\ 1 & a \text{ is student} \end{cases} \tag{45}$$

$$\epsilon = 1 \tag{46}$$

$$\zeta = 8 \tag{47}$$

$$\eta_a = \begin{cases} 4 & a \text{ is teacher} \\ 0 & a \text{ is student} \end{cases} \tag{48}$$

$$\theta = 1 \tag{49}$$

$$\iota = 8 \tag{50}$$

Notice that by far the biggest penalty comes from not assigning an event to a timeslot.

2.5 Complexity

Most common variants of non-trivial school timetabling problems have been proved to be $\mathcal{NP}$-hard, see Even et al. (1975); Cooper and Kingston (1996); ten Eikelder and Willemen (2001). However, we have not been able to find a formulation of timetabling which resemble exactly the one of this paper. Therefore it is proven in the following that both model (40) and the two-stage formulation is $\mathcal{NP}$-hard. This is done by using the well-known link between timetabling and graph-coloring.

2.5.1 HSTTP

To prove that HSTTP is $\mathcal{NP}$-hard, Proposition 6.3 of Wolsey (1998) is used. I.e. it must be shown that HSTTP is in $\mathcal{NP}$, and that another $\mathcal{NP}$-complete problem can be polynomially reduced to the decision-version of HSTTP (by Definition 6.7 of (Wolsey 1998, p. 88)).

Let the objective value of model (40) be denoted z_{HSTTP}. The decision version of model (40) asks whether a solution exists with objective $z_{\text{HSTTP}} \leq k$. Clearly the decision version of HSTTP is in $\mathcal{NP}$ as a solution $x_{e,r,t}$ to model (40), can be checked to have objective value less than k in polynomial time.

The well known $\mathcal{NP}$-complete problem Graph k-Colorability problem (GCP) asks whether it is possible to assign each vertex of a graph G a color such that no two adjacent vertices have the same color, using at most k colors. To show that GCP is polynomial reducible to HSTTP, a conversion scheme is now given, which transforms any instance of GCP into an instance of HSTTP. An instance of GCP consists of a graph G with vertices V and edges F, and a number of colors k.

- Start with an empty instance of HSTTP, i.e. $\mathcal{D} = \mathcal{M} = \mathcal{E} = \mathcal{C} = \emptyset$, $\mathcal{T} = \{t_D\}, \mathcal{R} = \{r_D\}$.
- For each vertex $v \in V$, create an event e.
- For each edge $f \in F$ between vertices v_1 and v_2, create an entity a. Let events e_1 and e_2 represent vertices v_1 and v_2, respectively. Assign entity a to both e_1 and e_2, i.e. $B_{e_1,a} = B_{e_2,a} = 1$.
- Create one day d with k timeslots $\{t_0, \ldots t_{k-1}\}$.
- Set $\alpha_{e,r,t} = \begin{cases} 0 & t = t_D \\ 1 & \text{else} \end{cases}$, and $\epsilon = \beta_a = \gamma_a = \delta_a = \zeta = \iota = \theta = \eta_a = 0$.
- As the only room in the instance is the dummy-room, the following substitution is made: $y_{e,t} = x_{e,r_D,t}$.

This problem-setting makes a lot of constraints and variables redundant. The HSTTP instance can therefore be written as follows (written as a maximization-problem by changing sign of $\alpha_{e,r,t}$):

HSTTP reduced problem (51)

$$\max \quad z_{\text{HSTTPReduced}} = \sum_{e,r,t} \alpha_{e,r,t} x_{e,r,t} \tag{51a}$$

s.t.

$$(\text{one time/room}) \sum_t x_{e,r_D,t} = 1 \qquad \forall e \tag{51b}$$

$$(\text{entity conf.}) \quad \sum_e B_{e,a} x_{e,r_D,t} \leq 1 \qquad \forall a, t \neq t_D \tag{51c}$$

Solving model (51) gives an objective $z_{\text{HSTTPReduced}}$, corresponding to the number of events which are assigned a timeslot different from the dummy-timeslot. By the conversion scheme, an event corresponds to a vertex in graph G and a timeslot corresponds to a color. Therefore $z_{\text{HSTTPReduced}}$ corresponds to the number of vertices which is assigned a color. To answer whether graph G is k-colorable, one can simply check if $z_{\text{HSTTPReduced}} = |V| \leq k$ (corresponding to solving the decision version of model (51)).

Any instance of GCP can thereby be converted into an instance of the decision version of HSTTP. Therefore the decision version of HSTTP is $\mathcal{NP}$-complete, so by Definition 6.7 of (Wolsey 1998, p. 88), HSTTP is $\mathcal{NP}$-hard.

2.5.2 Two-stage Formulation

By definition of stage one in the two-stage formulation, it differs from the original formulation by having only one room, the dummy-room. This is equivalent to the parameter setting in the proof of $\mathcal{NP}$-completeness for the original MIP. Therefore the exact same conversion scheme can be applied to this problem, and therefore this problem must be $\mathcal{NP}$-hard.

In the original description of the decomposition in Lach and Lübbecke (2012), stage two consisted of solving a bipartite perfect matching problem for each timeslot. However when taking soft-constraints such as room-stability into consideration, this problem structure is destroyed. Currently we have no insight on the complexity of the stage two problem, but the amount of time needed to solve the problem to optimality is in many cases negligible, as will be shown in Section 4. As stage one is $\mathcal{NP}$-hard, solving the two-stage formulation is $\mathcal{NP}$-hard.

3 Adaptive Large Neighborhood Search

In this section a heuristic solution approach based on ALNS is described.

Adaptive Large Neighborhood Search is a recent extension of the Large Neighborhood Search (LNS) paradigm, often credited to Ropke and Pisinger (2006). As in the LNS framework, first a destruct (ruin/remove) operator is applied to the solution at hand, and then a construct (recreate/insert) operator is used to repair the solution. In an ALNS framework, multiple destruct and construct operators are used, and the adaptive layer keeps track of their individual performance, and increases the probability of selecting operators which have previously performed 'good'. ALNS has mainly been applied to variants of the Vehicle Routing Problem (VRP) (Azi et al. (2010); Hemmelmayr et al. (2011); Salazar-Aguilar et al. (2011); Ribeiro and Laporte (2012)), but lately also other problem-domains (Muller et al. (2011); Muller (2010)).

We refer to Kristiansen et al. (2013), Kristiansen and Stidsen (2012) and Sørensen et al. (2012) (and references therein) for an introduction to ALNS for educational timetabling problems, and will here only briefly describe our implementation. This specific implementation is similar to that of Kristiansen et al. (2013), in which details such as method-selection and acceptance criteria are described. These are summarized below.

By description of LNS, one new solution S_{new} is found in each iteration. This solution is accepted with probability

$$\exp\left(\frac{z(S_{\text{cur}}) - z(S_{\text{new}})}{T}\right) \tag{52}$$

where $T \in \mathbb{R}^+$ is the current temperature, S_{cur} is the current solution, and denote by $z(S)$ the objective value of solution S. The initial temperature T_0 is selected by (S_0 denotes the initial solution)

$$T_0 = \frac{w_{\text{SA}} \cdot z(S_0)}{\ln 2} \tag{53}$$

and is decreased in each iteration by

$$T = d_{\mathrm{SA}} T \tag{54}$$

where $0 < d_{\mathrm{SA}} < 1$ is the decay factor.

A run of the algorithm is divided into segments $\{t_0, t_1, \dots, t_n\}$ each consisting of N_{it} iterations. Let π_i^t be the weight of heuristic i in segment t. Initially in the first section t_0, $\pi_i^{t_0} = 1 \ \forall i$. The probability of choosing heuristic i in segment t is $\frac{\pi_i^t}{\sum_j \pi_j^t}$. At the end of each segment t, the following update is performed for all heuristics,

$$\pi_i^{t+1} = \rho \frac{\bar{\pi}_i^t}{a_i^t} + (1-\rho)\pi_i^t \tag{55}$$

where a_i^t is the number of times heuristic i has been selected in segment t. $\bar{\pi}_i^t$ is the *observed* weight of heuristic i in segment t, which in each iteration is incremented depending on the quality of the new found solution. $\rho \in [0,1]$ is the reaction factor. A high reaction factor means that the weights of a segment will be very dependent upon the observed weights of the previous segment.

The observed weight $\bar{\pi}_i^t$ is updated in each iteration, by the following:

$$\mathrm{gap} = \frac{z(S_{\mathrm{cur}}) - z(S_{\mathrm{new}})}{z(S_{\mathrm{cur}})} \tag{56}$$

$$\bar{\pi}_i^t = \bar{\pi}_i^t + 5^{\min\left(\sigma \cdot \mathrm{gap}, 1\right)} \tag{57}$$

where $\sigma \in \mathbb{R}^+$ is the scaling parameter.

Hence this ALNS algorithm contains parameters w_{SA}, d_{SA} for controlling the acceptance-criteria, and parameters N_{it}, ρ, σ for controlling the adaptive selection of insert/remove methods.

A solution to the HSTTP is a list of (event,room,timeslot)-tuples, each representing an assignment of an event to a room and to a timeslot. The concept of a *move* is defined as follows: Given some feasible solution to an instance of the HSTTP, the move M permutes the solution S_1 such that a new feasible solution S_2 is obtained, and the change in the objective is $\Delta(M) = z(S_2) - z(S_1)$. For simplifying notation we only consider moves which does not yield an infeasible solution, however in practice such moves exist, but since they will never be applied to the solution in our implementation, they are ignored in this description. Four classes of moves have been implemented:

- $M_{ec,t}^{\mathrm{time}}$ assigns EventChain ec to timeslot t.
- $\cancel{M}_{ec,t}^{\mathrm{time}}$ un-assigns EventChain ec from timeslot t.
- $M_{e,r}^{\mathrm{room}}$ assigns event e to room r.
- $\cancel{M}_{e,r}^{\mathrm{room}}$ un-assigns event e from room r.

As the assign-time moves apply to EventChains, as opposed to events, the constraints for events which should be placed in the same/contiguous timeslots are handled implicitly, which simplifies the implementation.

By these moves, the remove- and insertion-operators are constructed.

3.1 Insertion methods

Algorithm 1 shows the general pseudo-code for the implemented insertion methods. M^{time} denotes a move which assigns an EventChain to a timeslot. The insertion methods differ in how they select this move in each iteration. Once an assign-time move has been selected, it is attempted to perform an assign-room move on each of the events in the EventChain of this assign-time move. See Algorithm 1.

Algorithm 1 Insertion method

```
input: A feasible solution S
loop
    M^time = ...
    if Δ(M^time) > 0 then
        return
    end if
    apply M^time to S
    for all e ∈ ec(M^time) do
        M^room = arg min_r Δ(M^room_{e,r})
        if Δ(M^room) < 0 then
            apply M^room to S
        end if
    end for
end loop
```

In the following, the approach for selecting the assign-time move is described for each insertion-method.

3.1.1 InsertGreedy

In each iteration, the move which reduces the objective most is chosen, written as

$$M^{\text{time}} = \arg\min_{ec,t} \Delta\left(M^{\text{time}}_{ec,t}\right) \tag{58}$$

3.1.2 InsertRegret-k

This is similar to the Regret-N neighborhood applied to variants of the VRP (Tillman and Cain (1972); Martello and Toth (1981); Potvin and Rousseau (1993)). Let $k \in \{2, 3, \ldots, |\mathcal{T}|\}$, and let $M^{\text{time}}_{ec,\{i\}}$ denote the i-th best time-move for EventChain ec. For a given k, the move selection is given by:

$$M^{\text{time}} = \underset{\substack{ec \\ \Delta\left(M^{\text{time}}_{ec,\{1\}}\right)<0}}{\arg\min} \left(\Delta\left(M^{\text{time}}_{ec,\{1\}}\right) - \sum_{i\geq 2}^{k} \Delta\left(M^{\text{time}}_{ec,\{i\}}\right) \right) \tag{59}$$

E.g. a InsertRegret-2 method selects in each iteration the best time move for the EventChain where the difference between the best time move and the second-best time move is most negative. The intuition is to perform the move which we will regret most if not done now. The following choices of k have been made by basic tests: 2, 3, 4, $|\mathcal{T}|$, which constitute four different insertion methods.

3.2 Remove methods

In each iteration of the ALNS, a number of events q is selected to be unassigned from timeslots in the solution at hand. The quantity q is selected as a random integer in the interval $\left[3, \max\left(p^{\text{des}}|\mathcal{E}|, 5\right)\right]$, where the parameter $p^{\text{des}} \in]0;1]$ describes the maximum percentage of events to be removed. As in Kristiansen et al. (2013), p^{des} is decreased with time, i.e. at time 0: $p^{\text{des}} = p^{\text{des}}_{\text{start}}$, and when reaching the timelimit: $p^{\text{des}} = p^{\text{des}}_{\text{end}}$. In between, a linear decay of the parameter is applied.

Algorithm 2 shows the general pseudo-code for the implemented destroy-methods. Let $RP()$ denote a remove procedure, which performs a number of unassign-moves and which returns the total number of events unassigned from timeslots. Recall that an event can not be assigned a room if it is not assigned a timeslot. Throughout this section it is therefore implicitly handled, that if an event is unassigned from a timeslot, it is also unassigned from its rooms.

Algorithm 2 Remove method

input: A feasible solution S, and remove-quantity q (number of events)
$\overline{q} = 0$
while $\overline{q} \leq q$ **do**
  $\overline{q} = \overline{q} + RP()$
end while

3.2.1 *RemoveRandom*

In this method, select a M randomly among all EventChains assigned a timeslot. This method will undoubtedly diversify the search. Let $E(M)$ denote the number of events of the EventChain of move M.

Algorithm 3 RP_{random}

input: A feasible solution S, and remove-quantity q
$\overline{q} = 0$
while $\overline{q} \leq q$ **do**
  Random select a move $\cancel{M}^{\text{time}}_{ec,t}$
  Apply $\cancel{M}^{\text{time}}_{ec,t}$ to S
  $\overline{q} = \overline{q} + E(\cancel{M}^{\text{time}}_{ec,t})$
end while
return $\overline{q}$

3.2.2 *RemoveRelated*

This method is related to Shaw operator (Shaw (1997, 1998)). The related measurement is defined as the percentage overlap among entities, and classes between two EventChains ec and ec', i.e.

$$\mathfrak{R}_{e,e'} = R_{\mathcal{A}} \frac{\left|\mathcal{A}(ec) \cup \mathcal{A}(ec')\right|}{\min(|\mathcal{A}(ec)|, |\mathcal{A}(ec')|)} + R_{\mathcal{C}} \frac{\left|\mathcal{C}(ec) \cup \mathcal{C}(ec')\right|}{\min(|\mathcal{C}(ec)|, |\mathcal{C}(ec')|)} \tag{60}$$

where $\mathcal{A}(ec)$ and $\mathcal{C}(ec)$ denote the set of entities and classes of EventChain ec, respectively. $R_{\mathcal{A}} \in [0,1]$ and $R_{\mathcal{C}} \in [0,1]$ are scaling parameters which require tuning. The amount of randomness in the selection is determined by p_{related}. Let the remove procedure be defined as follows:

Algorithm 4 $RP_{\text{RemoveRelated}}$

input: A feasible solution S, and remove-quantity q
$\overline{q} = 0$
ec = a random selected chain assigned to a timeslot
$D_{\text{done}} = \{ec\}$
while $\overline{q} < q$ **do**
  ec' = randomly selected from D_{done}
  L = all EventChains assigned to a timeslot, sorted by similarity to ec'
  Choose a random number $y \in [0;1[$
  ec = element number $y^{p_{\text{related}}}|L|$ of L
  Apply $\mathcal{M}^{\text{time}}_{ec,t}$ to S, where t is the timeslot assigned to EventChain ec
  $D_{\text{done}} = D_{\text{done}} \cup ec$
end while
return $\overline{q}$

3.2.3 *RemoveTime*

In this method, first select some random timeslot. Now remove EventChains assigned to this timeslot, until q EventChains have been removed. If at some point no more EventChains are assigned to the timeslot, select a new random timeslot. The exact remove-procedure looks as follows:

Algorithm 5 $RP_{\text{RemoveTime}}$

input: A feasible solution S, and remove-quantity q
$\overline{q} = 0$
$T_{\text{done}} = \emptyset$
while $\overline{q} < q$ **do**
  t = Randomly select from $\{\mathcal{T} \setminus T_{\text{done}}\}$
  while $\overline{q} < q$ **do**
    Randomly select an EventChain ec assigned to t
    Apply $\mathcal{M}^{\text{time}}_{ec,t}$ to S
    $\overline{q} = \overline{q} + E(\mathcal{M}^{\text{time}}_{ec,t})$
  end while
  $T_{\text{done}} = T_{\text{done}} \cup \{t\}$
end while
return $\overline{q}$

3.2.4 *RemoveClass*

Select a random class, and remove EventChains which contains it until q EventChains have been removed. If at some point no more EventChains are assigned to the timeslot, select a new random timeslot. The exact remove-procedure looks as follows:

Algorithm 6 $RP_{\text{RemoveClass}}$

input: A feasible solution S, and remove-quantity q
$\overline{q} = 0$
$C_{\text{done}} = \emptyset$
while $\overline{q} < q$ **do**
  $c =$ Randomly select from $\{\mathcal{C} \setminus C_{\text{done}}\}$
  while $\overline{q} < q$ **do**
    Randomly select an EventChain ec of c assigned to some timeslot t
    Apply $\cancel{M}^{\text{time}}_{ec,t}$ to S
    $\overline{q} = \overline{q} + E(\cancel{M}^{\text{time}}_{ec,t})$
  end while
  $C_{\text{done}} = C_{\text{done}} \cup \{c\}$
end while
return $\overline{q}$

3.3 Coupled destroy/repairs

Coupling certain destroy methods with certain repair methods is a small extension of the ALNS framework. This implies that the logic for choosing certain destroy/repair methods are extended, such that also certain pairs of methods can be chosen. This is useful for specialized destroy/repair methods, where a specific part of the solution is destroyed, and a competitive solution is not expected unless this part of the solution is repaired. In the following we describe a neighborhood where coupling seems useful, namely InsertRoom and RemoveRoom.

3.3.1 InsertRoom

In InsertRoom, rooms are assigned to events in a greedy way:

Algorithm 7 InsertRoom

input: A feasible solution S
loop
  $M^{\text{room}} = \arg\min_{e,r} \Delta(M^{\text{room}}_{e,r})$
  if $\Delta(M^{\text{room}}) > 0$ **then**
    `return`
  end if
  apply M^{room} to S
end loop

3.3.2 RemoveRoom

In RoomRemove, q random room-assignment are removed from the solution:

Algorithm 8 $RP_{\text{RemoveRoom}}$

input: A feasible solution S, and remove-quantity q
$\overline{q} = 0$
while $\overline{q} \leq q$ **do**
  Random select a move $\mathcal{M}^{\text{room}}_{e,r}$
  Apply $\mathcal{M}^{\text{room}}_{e,r}$ to S
  $\overline{q} = \overline{q} + E(\mathcal{M}^{\text{room}}_{e,r})$
end while
return $\overline{q}$

3.4 Parameter Tuning

A basic implementation of F-Race (Birattari (2005); Balaprakash et al. (2007)) is used to tune parameters for best algorithmic performance. The values obtained in Kristiansen et al. (2013) were used as a starting point. Table 1 shows the chosen value for each parameter.

Parameter	w_{SA}	d_{SA}	N_{it}	ρ	σ	$R_{\mathcal{A}}$	$R_{\mathcal{C}}$	p_{related}	$p^{\text{des}}_{\text{start}}$	$p^{\text{des}}_{\text{start}}$
Domain	$]0;1[$	$]0;1[$	$[0;\infty]$	$[0;1]$	$[0;\infty]$	$[0;1]$	$[0;1]$	$[1;\infty]$	$[0;1]$	$[0;1]$
Value	0.01	0.99	100	0.3	10000	0.7	0.3	20	0.10	0.01

Table 1: *List of parameters and their tuned value*

4 Results

The purpose of this section is to compare and evaluate the described solution approaches. A variety of datasets are therefore selected from the Lectio database. Using these datasets, we thereby aim at answering these question:

- How the found solutions compare with the bounds obtained from the IP-based solution approaches? This is a way of evaluating the quality of the obtained solutions, and/or the bounds obtained by the MIP approaches.
- Which solution approach obtains best solutions within a short timeframe? This is important to evaluate which approach to deploy to the users of Lectio.

4.1 Datasets

Currently the Lectio database contains almost 5000 potential datasets from 110 different high schools. Some of these datasets are obviously incomplete test-instances, in the sense that they lack information. It is attempted to filter out such instances; however, it can be hard to detect whether an instance is a 'test-instance'. Furthermore, datasets from the same school for the same year are considered identical. I.e. datasets are grouped by (school,year), and the most recent dataset from each group is chosen. A selection of 100 datasets is thereby made. Table 2 shows statistics for these datasets.

Table 2: *Statistics for selected datasets. Column '#ECs' denotes the number of EventChains, and the last two columns denote the amount of events which are part of an EventChain, and the event-to-timeslots ratio, respectively.*

Dataset	$\lvert\mathcal{E}\rvert$	#ECs	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{C}\rvert$	$\frac{\lvert\mathcal{E}\rvert}{\#\text{ECs}}$	$\frac{\lvert\mathcal{E}\rvert}{\lvert\mathcal{T}\rvert}$
AalborTG2012	768	296	116	5	7	35	231	202	0.39	21.9
AarhusA2011	768	50	101	5	6	30	383	306	0.07	25.6
AarhusA2012	803	44	94	5	6	30	437	302	0.05	26.8
Aars2009	682	10	47	5	9	45	158	209	0.01	15.2
Aars2010	1112	480	46	10	9	90	143	154	0.43	12.4
Aars2011	924	286	46	10	7	70	140	154	0.31	13.2
Aars2012	713	99	46	10	6	60	116	141	0.14	11.9
Alssund2010	729	293	38	5	9	45	214	229	0.40	16.2
Alssund2012	1473	3	34	10	9	90	215	245	0.00	16.4
BagsvaG2010	278	28	33	5	6	30	102	95	0.10	9.3
BirkerG2011	1637	193	81	5	9	45	587	439	0.12	36.4
BirkerG2012	1574	250	79	5	9	45	102	461	0.16	35.0
BjerrG2009	730	16	33	5	9	45	197	313	0.02	16.2
BjerrG2010	545	186	35	5	9	45	208	160	0.34	12.1
BjerrG2011	571	195	42	5	9	45	202	175	0.34	12.7
BjerrG2012	564	180	42	5	9	45	208	175	0.32	12.5
BroendG2012	389	38	31	10	4	40	74	102	0.10	9.7
CPHWGym2010	467	196	39	5	9	45	225	147	0.42	10.4
CPHWGym2011	511	223	39	5	9	45	245	165	0.44	11.4
CPHWGym2012	525	218	41	5	9	45	285	169	0.42	11.7
CPHWHG2012	634	6	27	5	8	40	217	163	0.01	15.9
CPHWHTX2010	530	28	35	5	10	50	111	124	0.05	10.6
CPHWHTX2011	688	213	34	5	10	50	110	121	0.31	13.8
CPHWHTX2012	434	13	30	5	10	50	89	82	0.03	8.7
DetFG2012	858	220	51	5	11	55	343	167	0.26	15.6
DetKG2010	185	8	26	5	7	35	111	69	0.04	5.3
DetKG2011	195	10	26	5	7	35	111	76	0.05	5.6
EUCN2009	249	18	55	5	7	35	89	78	0.07	7.1
EUCN2010	583	173	55	5	7	35	192	189	0.30	16.7
EUCN2011	286	108	64	5	7	35	29	98	0.38	8.2
EUCN2012	303	72	64	5	7	35	107	100	0.24	8.7
EUCNHG2010	190	83	29	5	8	40	49	56	0.44	4.8
EUCS2012	252	81	19	5	9	45	50	59	0.32	5.6
FaaborgG2008	1754	0	47	10	14	140	188	180	0.00	12.5
FalkonG2009	1139	2	71	10	5	50	468	326	0.00	22.8
FalkonG2011	955	1	72	10	5	50	369	284	0.00	19.1
FalkonG2012	1120	17	68	10	5	50	369	313	0.02	22.4
GUAasia2010	555	10	36	5	12	60	33	141	0.02	9.3
GUQaqor2011	524	262	15	10	10	100	90	70	0.50	5.2
GUQaqor2012	518	259	15	10	10	100	73	61	0.50	5.2
HadersK2011	662	3	75	5	5	25	483	318	0.00	26.5
HasserG2010	1275	75	71	10	5	50	526	384	0.06	25.5
HasserG2011	1369	86	79	10	5	50	585	408	0.06	27.4
HasserG2012	1471	146	70	10	5	50	623	422	0.10	29.4
HerningG2010	135	25	88	5	6	30	7	5	0.19	4.5
HerningG2011	1783	79	97	10	6	60	269	363	0.04	29.7
HerningG2012	1924	87	107	10	6	60	276	411	0.05	32.1
HoejeTaG2008	201	0	74	5	8	40	99	66	0.00	5.0
HoejeTaG2009	610	0	74	5	8	40	254	207	0.00	15.3
HoejeTaG2010	607	0	74	5	8	40	228	202	0.00	15.2
HoejeTaG2011	688	0	76	5	8	40	267	226	0.00	17.2
HoejeTaG2012	827	12	76	5	8	40	270	268	0.01	20.7
HorsenS2009	380	1	50	5	5	25	297	195	0.00	15.2
HorsenS2012	1119	5	54	10	5	50	551	409	0.00	22.4
Johann2012	1304	249	67	5	8	40	202	419	0.19	32.6
KalundG2011	1701	177	64	10	7	70	376	281	0.10	24.3
KalundG2012	1654	198	66	10	7	70	425	299	0.12	23.6
KalundHG2010	376	44	17	5	9	45	87	98	0.12	8.4
KoebenPG2012	95	1	22	5	6	30	63	43	0.01	3.2
KoegeH2012	1092	425	64	5	9	45	214	294	0.39	24.3
KongshoG2010	441	5	69	5	4	20	301	245	0.01	22.1
MariageG2009	692	10	71	10	4	40	240	183	0.01	17.3

Continued on next page

Table 2 – continued from previous page

Dataset	$\lvert\mathcal{E}\rvert$	#ECs	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{C}\rvert$	$\frac{\lvert\mathcal{E}\rvert}{\#\text{ECs}}$	$\frac{\lvert\mathcal{E}\rvert}{\lvert\mathcal{T}\rvert}$
MorsoeG2012	584	41	40	10	5	50	237	161	0.07	11.7
NaerumG2008	1533	0	75	10	4	40	567	513	0.00	38.3
NaerumG2009	1435	0	77	10	4	40	93	483	0.00	35.9
NielsSG2011	265	0	73	5	10	50	96	74	0.00	5.3
NielsSG2012	669	174	73	5	10	50	111	235	0.26	13.4
NordfynG2012	771	51	60	10	4	40	239	209	0.07	19.3
NyborgG2011	1191	4	59	10	5	50	466	336	0.00	23.8
OdderCfU2010	766	15	73	5	11	55	467	231	0.02	13.9
OdderG2009	782	2	47	10	6	60	175	199	0.00	13.0
OdderG2012	843	22	41	10	5	50	184	219	0.03	16.9
OrdrupG2010	1044	521	52	10	8	80	151	133	0.50	13.1
OrdrupG2011	1564	782	52	10	8	80	236	229	0.50	19.6
RibeK2011	837	2	61	5	8	40	263	227	0.00	20.9
RysenG2010	1477	36	74	10	4	40	396	319	0.02	36.9
RysenG2011	1294	26	74	10	4	40	427	395	0.02	32.4
RysenG2012	1382	59	75	10	4	40	469	516	0.04	34.6
SanktAG2012	773	16	47	10	4	40	63	210	0.02	19.3
SkanderG2010	1116	11	57	10	6	60	69	277	0.01	18.6
SkanderG2011	1161	14	56	10	6	60	463	284	0.01	19.4
SkanderG2012	1275	25	57	10	6	60	571	289	0.02	21.3
SkiveG2010	2665	1241	58	10	9	90	304	331	0.47	29.6
SlagelG2012	2152	164	103	10	6	60	607	469	0.08	35.9
SoendS2011	1206	76	98	10	4	40	383	332	0.06	30.2
SoendS2012	1278	1	111	10	4	40	340	379	0.00	32.0
StruerS2012	2915	160	71	10	9	90	348	401	0.05	32.4
VardeG2012	887	1	65	10	5	50	251	277	0.00	17.7
VejenG2009	928	10	52	10	6	60	209	189	0.01	15.5
Vejlefjo2011	714	63	45	5	14	70	186	234	0.09	10.2
VestfynG2009	585	234	64	5	8	40	260	168	0.40	14.6
VestfynG2010	582	239	62	5	8	40	246	167	0.41	14.6
VestfynG2011	619	255	57	5	8	40	257	180	0.41	15.5
VestfynG2012	613	254	51	5	8	40	195	180	0.41	15.3
ViborgK2011	1302	4	59	10	6	60	456	308	0.00	21.7
ViborgTG2009	549	90	27	5	10	50	87	120	0.16	11.0
ViborgTG2010	454	6	21	10	5	50	86	110	0.01	9.1
ViborgTG2011	473	42	23	10	4	40	90	109	0.09	11.8
VirumG2012	1731	225	65	10	4	40	536	443	0.13	43.3
VordingbG2009	615	57	67	5	7	35	262	227	0.09	17.6
Avg.	892	114	57	7	7	50	252	230	0.1	18.1
Max.	2915	1241	116	10	14	140	623	516	0.5	43.3

4.2 Solution approach comparison

In this following, a comparison between the proposed solution approaches is performed. In all cases Gurobi 5.01 has been used as MIP-solver, and tests were run in C# 5.0 using nUnit 2.6 on Windows 8 64bit. The machine was equipped with an Intel i7 CPU clocked at 2.80GHz and with 12GB of RAM. The default parameter settings of Gurobi were used. As initial solution ('MIPStart' parameter), events where assigning to either their locked timeslot/room or the dummy-timeslot/room. The percentage-gap between an objective value z and a lower bound LB is calculated by

$$\text{gap} = 100\frac{z - LB}{z} \tag{61}$$

The goal of this section is two-fold; 1) Compare the solutions obtained by each solution approach when applying a high time-limit. This is important for evaluating the potential of each approach. 2) Evaluate the solution approaches in terms of the bounds obtained by solving the MIP and the two-stage MIP. This is an important measure of solution quality.

The maximum number of threads and time-limits were set as follows:

	MIP	2-stage MIP	ALNS
Max. CPU threads	8	8	1
Time limit (s)	7200	6480 / 720	240

As we are interested in obtaining good bounds from the MIP approaches, these are allowed more computational time and more CPU threads. This means the comparison of solution quality favors the MIP approaches. In the next section, a more direct comparison of solution quality is made.

It should be noted that the described version of ALNS has no parallelization implemented, so it would not benefit from more threads. A parallel version is considered for future work (see e.g. Ropke (2009)).

Table 3 shows that the ALNS heuristic finds the best solution in 78 cases. In no cases are the pure MIP approach best, and in 20 cases are the two-stage approach best. In those cases where the two-stage approach is best, the dataset is usually small in terms of number of events. It was expected that the two-stage approach would outperform the pure MIP, but it is surprising that the ALNS heuristic outperforms both MIP approaches.

In total, a lower bound was found for 79 datasets. The MIP was able to find a lower bound in 46 cases, whereas the two-stage approach found a bound in 79 cases. This means that for the two-stage model, Gurobi was not able to solve the root relaxation of the stage one model in 21 cases, which is surprising. The model is not numerical instable, so currently our best guess is that we are facing issues with degeneracy. In 33 cases, the bound obtained by the MIP were best, and in 46 cases the bound obtained by the two-stage model were best. Note that if the root LP was not solved for a specific dataset, the reported solution is equal to the initial solution.

For those instances where a bound is found, the gap obtained for the ALNS is in average 25.6%, which seems rather high. Especially considering that those instances where a gap is not found are the big instances, where it seems likely that the gap is high. The inevitable question arises whether this is due to a poor bound, or due to poor solution quality. Future research will hopefully shed light upon this matter.

Figure 5a shows the linear regression of objectives as a function of number of events in dataset. This shows that as the size of datasets grows, the performance advantage of ALNS compared to the other solution approaches increases. I.e. the ALNS heuristic scales better with the size of the datasets. Figure 5b summarizes key measurements from Table 3.

Table 3: *Comparison of solution approaches. For the MIP model, column 'Time' shows the runtime, column 'Obj' shows the objective of the obtained solution, column 'LB' shows the lower bound, and 'Gap' shows the gap between the objective and the bound found by the MIP and the two-stage MIP. For the two-stage MIP, column 'Stg1' and 'Stg2' shows the elapsed time for solving the first and second stage, respectively. The remaining columns are analogous to the those defined for the MIP approach. For ALNS is shown the average objective found over 10 runs 'Obj', the standard deviation of these runs 'σ', and 'Gap' denotes the gap between the average objective and the best bound found. When a bound or solution is not found within the timelimit, a dash is written. The best solution for each dataset is written in* ***bold font*** *(skipping draws).*

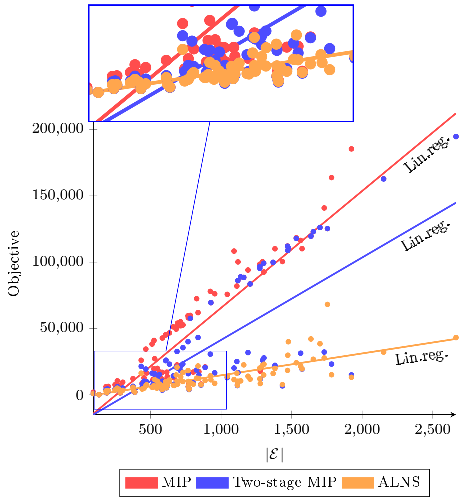
	MIP				Two-stage MIP					ALNS		
Dataset	Time	Obj	LB	Gap	Stg1	Stg2	Obj	LB	Gap	Obj	σ	Gap
AalborTG2012	>7200	6118	5946	2.8	>6480	1	**6018**	5934	1.2	6317	66.6	5.9
AarhusA2011	>7200	58015	-	89.7	>6480	93	15872	5986	62.3	**10037**	387.2	40.4
AarhusA2012	>7200	17096	5722	64.9	>6480	>720	8947	6005	32.9	**7971**	87.9	24.7
Aars2009	>7200	49504	-	76	>6480	6	20780	11874	42.9	**14900**	154	20.3
Aars2010	>7200	81970	-	84	>6480	15	25057	13134	47.6	**16268**	158.8	19.3
Aars2011	>7200	77967	-	87.6	>6480	11	30623	9709	68.3	**14256**	287.1	31.9
Aars2012	>7200	55049	-	86.5	>6480	4	21206	7456	64.8	**10701**	99.1	30.3
Alssund2010	>7200	52717	-	87.1	>6480	27	23173	6811	70.6	**9967**	438.5	31.7
Alssund2012	>7200	108810	-	-	>6480	2	108810	-	-	**29803**	609.9	-
BagsvaG2010	>7200	6777	3171	53.2	>6480	14	**3916**	3063	19	3960	87.8	19.9
BirkerG2011	>7200	119600	-	-	>6480	1	119600	-	-	**42063**	751.9	-
BirkerG2012	>7200	110180	-	85.8	>6480	>720	**19322**	15662	18.9	19552	54.1	19.9
BjerrG2009	>7200	52639	-	78.9	>6480	8	35514	11094	68.8	**16877**	271	34.3
BjerrG2010	>7200	12868	3928	69.5	>6480	22	5788	3868	32.1	**4983**	74	21.2
BjerrG2011	>7200	13009	4142	68.2	>6480	>720	9302	4060	55.5	**6334**	119.1	34.6
BjerrG2012	>7200	17200	5055	70.6	>6480	354	15265	5007	66.9	**8023**	220.3	37
BroendG2012	>7200	2005	1881	6.2	1173	14	**1929**	1859	2.5	2040	30	7.8
CPHWGym2010	>7200	34415	-	89.1	>6480	2	19363	3759	80.6	**6775**	328.6	44.5
CPHWGym2011	>7200	38232	-	89.3	>6480	2	16212	4095	74.7	**5679**	179.3	27.9
CPHWGym2012	>7200	40945	-	89.7	>6480	2	15543	4205	73	**6762**	217.7	37.8
CPHWHG2012	>7200	46625	8157	82.1	>6480	10	23088	8338	63.9	**11077**	227.6	24.7
CPHWHTX2010	>7200	27174	9179	66.2	>6480	10	15943	8828	42.4	**11342**	146.4	19.1
CPHWHTX2011	>7200	22466	20460	8.9	>6480	5	**20708**	18490	1.2	20734	27.6	1.3
CPHWHTX2012	>7200	25998	14481	44.3	>6480	13	21392	13115	32.3	**16256**	126.1	10.9
DetFG2012	>7200	8017	7168	10.6	>6480	6	**7265**	7018	1.3	7560	73.4	5.2
DetKG2010	>7200	6058	1732	69.9	>6480	1	4006	1821	54.5	**2947**	69	38.2
DetKG2011	>7200	5594	1732	68.2	>6480	14	4366	1780	59.2	**2820**	136	36.9
EUCN2009	>7200	7557	2911	61.5	>6480	2	4298	2856	32.3	**3737**	116	22.1
EUCN2010	>7200	4231	3329	21.3	>6480	32	**3463**	3246	3.9	3882	76	14.2
EUCN2011	>7200	1435	1395	2.8	>6480	1	**1430**	1384	2.5	1468	13	4.9
EUCN2012	>7200	9430	2327	74.9	>6480	1	5059	2363	53.3	**3289**	160	28.2
EUCNHG2010	>7200	1476	1371	7.1	>6480	5	**1421**	1368	3.5	1505	28	8.9
EUCS2012	>7200	4689	3576	23.7	>6480	12	3783	3347	5.5	**3714**	32	3.7
FaaborgG2008	>7200	125330	-	-	>6480	54	125330	-	-	**68124**	2156	-
FalkonG2009	>7200	88890	-	-	>6480	0	88890	-	-	**10449**	251	-
FalkonG2011	>7200	76170	-	93.2	>6480	>720	16543	5183	68.7	**8584**	271	39.6
FalkonG2012	>7200	100190	-	93.9	>6480	>720	16666	6105	63.4	**10143**	432	39.8
GUAasia2010	>7200	6579	6354	3.4	6	>720	**6461**	6035	1.7	6527	7	2.7
GUQaqor2011	>7200	19623	4537	76.8	>6480	6	10005	4554	54.5	**6674**	301	31.8
GUQaqor2012	>7200	11488	4314	62.5	>6480	15	7619	4294	43.4	**5733**	134	24.8
HadersK2011	>7200	51190	-	92.4	>6480	>720	14229	3909	72.5	**7128**	386	45.2
HasserG2010	>7200	96790	-	-	>6480	0	96790	-	-	**11963**	132	-
HasserG2011	>7200	99840	-	-	>6480	1	99840	-	-	**16061**	472	-
HasserG2012	>7200	112160	-	-	>6480	2	112034	-	-	**18338**	672	-
HerningG2010	2	37	37	0	0	2	37	35	0	37	0	0
HerningG2011	>7200	163785	-	94	>6480	12	23117	9829	57.5	**15091**	144	34.9
HerningG2012	>7200	185433	-	94.7	>6480	>720	14952	9763	34.7	**13147**	76	25.7
HoejeTaG2008	>7200	6292	2253	59.3	>6480	1	**2707**	2563	5.3	2958	92	13.4
HoejeTaG2009	>7200	45260	-	87.2	>6480	470	26066	5773	77.9	**9157**	303	37

Continued on next page

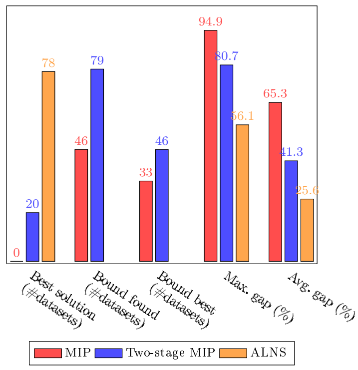
Table 3 – continued from previous page

	MIP				Two-stage MIP					ALNS		
Dataset	Time	Obj	LB	Gap	Stg1	Stg2	Obj	LB	Gap	Obj	σ	Gap
HoejeTaG2010	>7200	45095	-	86.3	>6480	116	25678	6188	75.9	**9862**	232	37.3
HoejeTaG2011	>7200	51050	-	86.8	>6480	48	32630	6726	79.4	**10158**	201.5	33.8
HoejeTaG2012	>7200	72455	7592	89.2	>6480	224	18627	7845	57.9	**12502**	143.4	37.3
HorsenS2009	63	3100	3100	0	0	2	3100	2865	0	3111	9	0.4
HorsenS2012	>7200	86090	-	-	>6480	0	86090	-	-	**10056**	434.9	-
Johann2012	>7200	92575	-	80.1	>6480	>720	27781	18456	33.6	**23001**	193.4	19.8
KalundG2011	>7200	126150	-	-	>6480	1	126150	-	-	**38479**	514.5	-
KalundG2012	>7200	123010	-	-	>6480	1	123010	-	-	**26768**	348.8	-
KalundHG2010	>7200	12103	4540	62.4	>6480	16	6351	4551	28.3	**5631**	83.1	19.2
KoebenPG2012	>7200	1872	637	65.7	>6480	2	**874**	642	26.5	888	31	27.7
KoegeH2012	>7200	108347	-	91.6	>6480	2	20150	9096	54.9	**11418**	136.2	20.3
KongshoG2010	>7200	8889	2411	72	>6480	3	7954	2488	68.7	**4296**	175.8	42.1
MariageG2009	>7200	54030	-	90.5	>6480	161	20138	5118	74.6	**8013**	251.6	36.1
MorsoeG2012	>7200	42762	-	91	>6480	54	10241	3854	62.4	**5651**	122.6	31.8
NaerumG2008	>7200	118370	-	-	>6480	0	117894	-	-	**24104**	502.8	-
NaerumG2009	>7200	100450	-	94.9	209	>720	**6681**	5114	23.5	7667	62.5	33.3
NielsSG2011	>7200	10464	3323	67.4	>6480	2	6132	3412	44.4	**4953**	111.7	31.1
NielsSG2012	>7200	12747	5722	55	>6480	16	8003	5738	28.3	**6952**	107.4	17.5
NordfynG2012	>7200	8201	4152	49.4	>6480	205	**4890**	4048	15.1	5160	38.6	19.5
NyborgG2011	>7200	94059	-	93.5	>6480	>720	31809	6129	80.7	**13944**	434.4	56.1
OdderCfU2010	>7200	59540	-	79.5	>6480	1	40032	12188	69.6	**18219**	189.2	33.1
OdderG2009	>7200	59851	-	-	>6480	2	57586	-	-	**9308**	206.8	-
OdderG2012	>7200	17402	9602	44.8	>6480	>720	14888	8878	35.5	**12307**	157.8	22
OrdrupG2010	>7200	75700	-	85.9	>6480	37	**12936**	10665	17.6	13663	391.8	21.9
OrdrupG2011	>7200	116400	-	85.5	>6480	>720	31329	16904	46	**21612**	630.4	21.8
RibeK2011	>7200	61945	-	73.8	>6480	390	43175	16209	62.5	**21679**	260.1	25.2
RysenG2010	>7200	110690	-	-	>6480	1	110690	-	-	**39971**	148	-
RysenG2011	>7200	100313	-	82.3	>6480	>720	25989	17756	31.7	**22260**	99.1	20.2
RysenG2012	>7200	110111	-	86.3	>6480	>720	22156	15115	31.8	**19841**	189.2	23.8
SanktAG2012	>7200	4624	3415	26.2	75	>720	**3911**	3376	12.7	4207	33.5	18.8
SkanderG2010	>7200	7708	6051	21.5	77	>720	**6875**	5712	12	7209	36.5	16.1
SkanderG2011	>7200	88470	-	-	>6480	0	88470	-	-	**22525**	368.5	-
SkanderG2012	>7200	98487	-	-	>6480	1	95319	-	-	**20138**	682.8	-
SkiveG2010	>7200	194740	-	-	>6480	21	194740	-	-	**43120**	1261.2	-
SlagelG2012	>7200	162960	-	-	>6480	36	162765	-	-	**32167**	1225.7	-
SoendS2011	>7200	83560	-	-	>6480	1	83560	-	-	**11776**	248.4	-
SoendS2012	>7200	17778	6838	61.5	>6480	2	11915	6647	42.6	**8420**	94	18.8
StruerS2012	>7200	-	-	-	>6480	18	207488	-	-	**73361**	3188	-
VardeG2012	>7200	20933	5921	71.7	>6480	13	20622	5720	71.3	**10777**	1911	45.1
VejenG2009	>7200	69450	-	-	>6480	0	69450	-	-	**11264**	209	-
Vejlefjo2011	>7200	52035	-	83.6	>6480	>720	18043	8511	52.8	**13514**	183	37
VestfynG2009	>7200	11606	4176	64	>6480	347	5999	4137	30.4	**5973**	148.5	30.1
VestfynG2010	>7200	16895	4308	74.5	>6480	354	**5974**	4225	27.9	6761	211.3	36.3
VestfynG2011	>7200	13624	5110	62.5	>6480	25	**6657**	4925	23.2	7013	218.7	27.1
VestfynG2012	>7200	11095	4279	61.4	>6480	48	**5212**	4210	17.9	5244	52.5	18.4
ViborgK2011	>7200	99170	-	-	>6480	0	99170	-	-	**14923**	406.1	-
ViborgTG2009	>7200	19891	8695	56.3	>6480	33	12077	8356	28	**10216**	102	14.9
ViborgTG2010	>7200	12727	4130	67.6	>6480	112	10226	3990	59.6	**4932**	66	16.3
ViborgTG2011	>7200	16433	6716	59.1	>6480	46	9808	6204	31.5	**7478**	40	10.2
VirumG2012	>7200	140883	-	87.4	>6480	>720	32183	17770	44.8	**27738**	502	35.9
VordingbG2009	>7200	17025	5457	68	>6480	91	9905	5243	44.9	**8568**	97	36.3
Avg.†				65.3					41.3			25.6
Max.				94.9					80.7			56.1
No. times best		0					20			78		
No. bound found			46					79				
No. best bound			33					46				

† Rows where either of the gap columns are not available are skipped for a fair comparison.



(a) *Objective of datasets as a function of number of events. Linear regression for each set of points is also shown.*



(b) *Bar plot summary from Table 3.*

Fig. 5: *Performance of the three solution approaches illustrated.*

4.3 Operational considerations

In this section it is considered what approach can be used to find the best solutions within a short time horizon. I.e. for the MIP approaches, we are not interested in the bound, but only in actual solutions. According to the documentation of Gurobi, some parameters will make the solver focus on finding good solutions. Table 4 shows the chosen values for these parameters. Furthermore the max. number of threads for Gurobi is set to 1, to obtain a fair comparison with the ALNS heuristic.

Table 4: *Gurobi parameter settings in operational setting*

	Default	Value	Intention
MIPFocus	0	1	Focus on finding good solutions
Heuristics	0.05	0.90	Attempt to spend 90% of solver time on heuristics
ImproveStartTime	infinity	0	Immediately focus on solution quality
Cuts	-1	0	Turn off all cuts

Table 5 shows for each dataset the best found solution by each solution approach after 240, 420 and 600 seconds. A user of Lectio will usually not run the algorithm for more than 10 minutes, so these time horizons seem appropriate.

The table shows that the ALNS algorithm finds better solutions for all three time horizons in the majority of cases (88, 86 and 87, respectively). Furthermore ALNS is able to find a feasible solution in all cases. This clearly makes ALNS the strongest candidate to deploy to the users of Lectio.

Table 5: *Best found solutions for the three solution approaches at certain points in time.*

	240s			420s			600s		
Dataset	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS
AalborTG2012	8530	**6182**	6296	8530	**6069**	6278	7776	**6067**	6259
AarhusA2011	58015	58015	**10244**	58015	58015	**10063**	58015	58015	**9995**
AarhusA2012	66350	12121	**8013**	66350	10966	**7907**	66350	10738	**7880**
Aars2009	-	49484	**15068**	49504	49484	**14900**	49504	49484	**14835**
Aars2010	-	81970	**16474**	-	30344	**16250**	-	30344	**16127**
Aars2011	-	64774	**14511**	77967	64774	**14230**	77967	64774	**14150**
Aars2012	-	52520	**10674**	55049	25984	**10544**	55049	25252	**10510**
Alssund2010	52717	52585	**9825**	52717	52585	**9714**	52717	52585	**9658**
Alssund2012	-	108810	**30349**	-	108810	**29088**	-	108810	**28530**
BagsvaG2010	9212	5855	**3942**	9212	5400	**3939**	8142	5247	**3939**
BirkerG2011	-	119600	**42190**	-	119600	**41415**	119600	119600	**41066**
BirkerG2012	-	**19526**	19642	-	**19497**	19566	-	19464	**19434**
BjerrG2009	52639	52395	**17152**	52639	52395	**16846**	52639	52395	**16716**
BjerrG2010	44901	7327	**4923**	14512	7118	**4889**	14512	7052	**4885**
BjerrG2011	52933	11371	**6339**	52933	11059	**6302**	52933	10796	**6257**
BjerrG2012	39745	39745	**7974**	39745	39745	**7838**	39745	39745	**7797**
BroendG2012	2263	**1935**	2040	2207	**1935**	2036	2002	**1935**	2036
CPHWGym2010	34415	34415	**6670**	34415	34415	**6583**	34415	34415	**6560**
CPHWGym2011	38232	37368	**5713**	38232	16573	**5628**	38232	16573	**5601**
CPHWGym2012	40945	37095	**6554**	40945	37095	**6508**	40945	21405	**6487**
CPHWHG2012	46625	46099	**11023**	46625	46099	**10954**	46625	46099	**10919**
CPHWHTX2010	38497	21441	**11293**	38497	21441	**11215**	38497	21441	**11199**
CPHWHTX2011	30578	21508	**20745**	24403	21349	**20729**	24403	21024	**20724**
CPHWHTX2012	31860	27169	**16137**	31860	26028	**16096**	31860	25441	**16090**
DetFG2012	19736	**7460**	7583	14147	**7417**	7557	13618	**7415**	7546
DetKG2010	7556	6139	**2951**	7556	6046	**2949**	7418	6041	**2949**
DetKG2011	14445	6360	**2863**	6653	6348	**2848**	6653	6185	**2846**
EUCN2009	18590	5833	**3680**	18590	5693	**3672**	7856	5335	**3671**
EUCN2010	5958	3961	**3940**	5947	**3873**	3916	5947	**3873**	3910
EUCN2011	**1451**	1484	1479	**1448**	1478	1479	**1448**	1478	1479
EUCN2012	22570	8262	**3253**	22570	7549	**3227**	22570	6993	**3223**
EUCNHG2010	1847	**1443**	1488	1842	**1442**	1484	1842	**1442**	1484
EUCS2012	6629	4229	**3710**	5169	4078	**3707**	5169	4078	**3706**
FaaborgG2008	-	125330	**69675**	-	125330	**64777**	-	125330	**62082**
FalkonG2009	-	88890	**10541**	-	88890	**10212**	-	88890	**10098**
FalkonG2011	-	76170	**8629**	76170	76170	**8375**	76170	76170	**8286**
FalkonG2012	-	82629	**10153**	100190	82629	**9786**	100190	82629	**9688**
GUAasia2010	-	**6466**	6534	38850	**6462**	6515	38850	**6457**	6509
GUQaqor2011	38210	11910	**6749**	38210	11795	**6625**	38210	11810	**6577**
GUQaqor2012	42346	10810	**5780**	42346	9306	**5694**	42346	9296	**5666**
HadersK2011	51190	51190	**7156**	51190	51190	**7029**	51190	51190	**6974**
HasserG2010	-	96790	**12061**	96790	96790	**11645**	96790	96790	**11491**
HasserG2011	-	99840	**15970**	-	99840	**15479**	99840	99840	**15280**
HasserG2012	-	112034	**19099**	-	112034	**18379**	-	112034	**18016**
HerningG2010	-	**37**	37	-	**37**	37	-	**37**	37
HerningG2011	163785	102119	**15971**	163785	26648	**15602**	163785	26646	**15347**
HerningG2012	185433	-	**13290**	185433	28140	**13117**	185433	28140	**12964**
HoejeTaG2008	14865	4950	**2941**	6923	4646	**2921**	6923	4175	**2920**
HoejeTaG2009	-	45260	**9275**	45260	45260	**9118**	45260	28064	**9079**
HoejeTaG2010	-	45095	**9733**	45095	32686	**9644**	45095	28120	**9615**
HoejeTaG2011	-	51050	**10168**	-	51050	**10065**	51050	51050	**10021**
HoejeTaG2012	-	30074	**12468**	-	30074	**12349**	-	25206	**12298**
HorsenS2009	3100	**3100**	3115	3100	**3100**	3115	3100	**3100**	3115
HorsenS2012	-	86090	**10181**	-	86090	**9733**	86090	86090	**9550**
Johann2012	-	92575	**23104**	-	31672	**22892**	92575	31507	**22780**
KalundG2011	-	126150	**39102**	-	126150	**38464**	126150	126150	**37929**
KalundG2012	-	123010	**27503**	-	123010	**26451**	-	123010	**25828**
KalundHG2010	27530	7981	**5631**	12214	7879	**5599**	12008	7758	**5596**
KoebenPG2012	2517	1589	**876**	2517	1483	**876**	2517	1333	**876**
KoegeH2012	-	26279	**11431**	108347	22745	**11338**	108347	21005	**11274**
KongshoG2010	34265	10249	**4302**	34265	8675	**4278**	34265	8602	**4268**
MariageG2009	54030	54030	**8152**	54030	54030	**8045**	54030	54030	**7962**

Continued on next page

Table 5 – continued from previous page

	240s			420s			600s		
Dataset	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS
MorsoeG2012	42762	42395	**5681**	42762	42395	**5627**	42762	42395	**5609**
NaerumG2008	-	117894	**24543**	118370	117894	**23776**	118370	117894	**23311**
NaerumG2009	-	**6752**	7767	-	**6752**	7679	-	**6746**	7545
NielsSG2011	-	10828	**4852**	19200	10486	**4799**	19200	8954	**4796**
NielsSG2012	50253	11041	**6945**	50253	10756	**6902**	50253	10752	**6848**
NordfynG2012	63210	6216	**5218**	63210	5909	**5158**	8227	5909	**5137**
NyborgG2011	-	85372	**14041**	-	85372	**13647**	94059	85372	**13430**
OdderCfU2010	-	59473	**18229**	-	59473	**18059**	-	59473	**17997**
OdderG2009	59851	57586	**9271**	59851	57586	**9037**	59851	57586	**8939**
OdderG2012	86520	16963	**12482**	86520	14977	**12282**	86520	14976	**12210**
OrdrupG2010	75700	75700	**13645**	75700	14806	**13465**	75700	13944	**13337**
OrdrupG2011	-	116400	**21986**	-	116400	**21798**	-	116400	**21541**
RibeK2011	-	61945	**21762**	61945	61945	**21544**	61945	61945	**21490**
RysenG2010	-	110690	**40194**	-	110690	**40010**	110690	110690	**39778**
RysenG2011	-	89741	**22391**	100313	89741	**22191**	100313	89741	**22006**
RysenG2012	-	95590	**20242**	-	95590	**19910**	110111	95590	**19598**
SanktAG2012	54080	**3824**	4252	54080	**3821**	4200	54080	**3819**	4172
SkanderG2010	-	**6893**	7320	-	**6878**	7234	-	**6876**	7152
SkanderG2011	-	88470	**22985**	-	88470	**22374**	-	88470	**22087**
SkanderG2012	-	95319	**20367**	-	95319	**19659**	-	95319	**19334**
SkiveG2010	-	194740	**43699**	-	194740	**42967**	-	194740	**42127**
SlagelG2012	-	162765	**30743**	-	162765	**30186**	-	162765	**29673**
SoendS2011	83560	83560	**12049**	83560	83560	**11674**	83560	83560	**11519**
SoendS2012	87883	14589	**8451**	16717	13920	**8355**	16717	12765	**8298**
StruerS2012	-	207488	**69927**	-	207488	**68367**	-	207488	**67294**
VardeG2012	60980	60980	**9684**	60980	60980	**9526**	60980	60980	**9455**
VejenG2009	-	69450	**11224**	-	69450	**10886**	69450	69450	**10779**
Vejlefjo2011	-	52035	**13478**	-	52035	**13338**	52035	52035	**13293**
VestfynG2009	62063	7914	**6117**	62063	**5867**	6037	62063	**5755**	6011
VestfynG2010	61216	11853	**6647**	61216	10155	**6562**	61216	9303	**6548**
VestfynG2011	67790	8577	**7068**	67790	8521	**6997**	67790	8269	**6936**
VestfynG2012	66096	7607	**5241**	66096	7354	**5187**	66096	7352	**5182**
ViborgK2011	-	99170	**15459**	-	99170	**14670**	-	99170	**14360**
ViborgTG2009	39385	21077	**10229**	39385	14309	**10156**	39385	13630	**10141**
ViborgTG2010	34980	12299	**4972**	34980	11733	**4943**	34980	10977	**4934**
ViborgTG2011	36300	11887	**7496**	36300	9723	**7474**	36300	9723	**7469**
VirumG2012	-	111119	**23561**	140883	111119	**23359**	140883	34158	**23176**
VordingbG2009	55115	11646	**8607**	55115	10953	**8535**	55115	10939	**8523**
No. solutions	55	99	100	70	100	100	81	100	100
No. times best	1	11	88	1	13	86	1	12	87

5 Conclusion

A complex model of timetabling for high schools in Denmark has been described. This model is build upon the timetabling component of Lectio, and has been proven to be $\mathcal{NP}$-hard. An in-depth description of a MIP approach has been given, and a simple decomposition suggested. Furthermore, a heuristic based on Adaptive Large Neighborhood Search has been discussed, which yields a total of three different solution approaches.

Using 100 real-life datasets, these solution approaches have been evaluated in using lower bounds and solution quality in a production setting. The ALNS heuristic proved to perform best wrt. both aspects.

The gap between the solutions found by ALNS and the best bound found is in average 25.6%, which is unsatisfactory. Future research will hopefully be able to narrow this gap, either by finding better bounds, or by strengthening the solution approaches.

The first version of the ALNS heuristic went into production in Lectio on 27/2-2012. The response received from the high schools has been positive, and numerous feature requests and problem extensions have been suggested. This facilitates the ongoing development of the algorithms.

The chosen problem formulation might leave events which are not assigned to a timeslot. This can happen either because the problem is too constrained due to the parameters set by the high school, or it can happen as a sub-optimal solution was provided by the algorithm. A way to tackle such unassigned events is topic for future work. Currently we believe a solution could be to incorporate a different algorithm which is independent of the solution approaches described in this paper. I.e. once the algorithm is finished, assume a number of events are left unassigned to timeslots. Now the user of Lectio can attempt to find a timeslot for one or several of these events, using an algorithm which slightly permutes the timetable, and thereby finds reasonable timeslots for the selected unassigned events. This algorithm could possible be based on a concept like *Cyclic Transfers* (Post et al. (2010)) or a variant of the *Repair Problem* for timetables (Kaneko et al. (1999); Kingston (2012)).

References

J. S. Appleby, D. V. Blake, and E. A. Newman. Techniques for producing school timetables on a computer and their application to other scheduling problems. *The Computer Journal*, 3 (4):237–245, 1961. doi: 10.1093/comjnl/3.4.237.

A. Asratian and D. de Werra. A generalized class-teacher model for some timetabling problems. *European Journal of Operational Research*, 143(3):531 – 542, 2002. ISSN 0377-2217. doi: 10.1016/S0377-2217(01)00342-3.

P. Avella and I. Vasil'Ev. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8:497–514, 2005. ISSN 1094-6136. 10.1007/s10951-005-4780-1.

P. Avella, B. DAuria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231. 10.1007/s10732-007-9025-3.

N. Azi, M. Gendreau, and J.-Y. Potvin. *An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips.* CIRRELT, 2010.

P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In *Proceedings of the 4th international conference on Hybrid metaheuristics*, HM'07, pages 108–122, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-75513-6, 978-3-540-75513-5.

V. Bardadym. Computer-aided school and university timetabling: The new wave. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 22–45. Springer Berlin / Heidelberg, 1996.

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

R. E. Bixby. *Optimization Stories*, volume Extra of *21st International Symposium on Mathematical Programming Berlin*, chapter A Brief History of Linear and Mixed-Integer Programming Computation, pages 107–121. Journal der Deutschen Mathematiker-Vereinigung, August 19–24 2012.

A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, pages 1–12, 2010. ISSN 0254-5330. 10.1007/s10479-010-0707-0.

E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266 – 280, 2002. ISSN 0377-2217. doi: 10.1016/S0377-2217(02)00069-3.

E. Burke, J. Kingston, and P. Pepper. A standard data format for timetabling instances. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 213–222. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0055891.

E. Burke, J. Marecek, A. Parkes, and H. Rudova. A branch-and-cut procedure for the udine course timetabling problem. *Annals of Operations Research*, pages 1–17, 2008. ISSN 0254-5330. 10.1007/s10479-010-0828-5.

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

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

P. D. Causmaecker and G. Berghe. Towards a reference model for timetabling and rostering. *Annals of Operations Research*, pages 1–10, 2010. ISSN 0254-5330. 10.1007/s10479-010-0721-2.

T. Cooper and J. Kingston. The complexity of timetable construction problems. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 281–295. Springer Berlin / Heidelberg, 1996.

S. Daskalaki, T. Birbas, and E. Housos. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153:117–135, 2004.

P. Dige, C. Lund, and H. Raun. Timetabling by simulated annealing applied simulated annealing. *Lecture Notes in Economics and Mathematical Science*, 396:151–174, 1993.

S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:184–193, 1975. ISSN 0272-5428.

G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza. A sa-ils approach for the high school timetabling problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

L. D. Gaspero, A. Schaerf, and B. McCollum. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queen's University SARC Building, Belfast, United Kingdom, 2007.

M. Gendreau and E. Burke, editors. *PATAT2008: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, 18 - 22 August 2008.

C. C. Gotlieb. The construction of class-teacher timetables. In C. M. Popplewell, editor, *IFIP Congress*, volume 62, pages 73–77, North-Holland Pub. Co, 1962.

V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Technical Report CIRRELT-2011-42, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, July 2011.

K. Kaneko, M. Yoshikawa, and Y. Nakakuki. Improving a heuristic repair method for large-scale school timetabling problems. In J. Jaffar, editor, *Principles and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*, pages 275–288. Springer Berlin / Heidelberg, 1999. ISBN 978-3-540-66626-4.

A. Kheiri, E. Ozcan, and A. J. Parkes. Hysst: Hyper-heuristic search strategies and timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

J. Kingston. The kts high school timetabling system. In E. Burke and H. RudovÃ¡, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 308–323. Springer Berlin / Heidelberg, 2007.

J. Kingston. Resource assignment in high school timetabling. *Annals of Operations Research*, pages 1–14, 2010. ISSN 0254-5330. 10.1007/s10479-010-0695-0.

J. H. Kingston. Repairing high school timetables with polymorphic ejection chains. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

D. Kjenstad, A. Riise, T. E. Nordlander, B. McCollum, and E. Burke, editors. *PATAT 2012: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, 29 -31 August 2012.

S. Kristiansen and T. R. Stidsen. Adaptive large neighborhood search for student sectioning at danish high schools. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

S. Kristiansen, M. Sørensen, T. Stidsen, and M. Herold. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, To appear, 2013.

G. Lach and M. Lübbecke. Optimal university course timetables and the partial transversal polytope. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 235–248. Springer Berlin / Heidelberg, 2008.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330. 10.1007/s10479-010-0700-7.

N. L. Lawrie. An integer linear programming model of a school timetabling problem. *The Computer Journal*, 12(4):307–316, 1969.

R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30:167–190, 2008. ISSN 0171-6468. 10.1007/s00291-007-0097-0.

R. Lewis, B. Paechter, and B. McCollum. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff Accounting and Finance Working Papers A2007/3, Cardiff University, Cardiff Business School, Accounting and Finance Section, 2007.

M. Lundberg-Jensen, J. Bruun, and J. Ahmt. Task assignment for high school teachers. Technical report, Operations Management, Department of Informatics and Mathematical Modeling. Kgs. Lyngby. Technical University of Denmark, 2008.

S. Martello and P. Toth. An algorithm for the generalized assignment problem. *Operational research*, 81:589–603, 1981.

C. H. Martin. Ohio university's college of business uses integer programming to schedule classes. *Interfaces*, 34(6):460–465, November 2004. doi: 10.1287/inte.1040.0106.

B. McCollum, E. Burke, and G. White, editors. *PATAT2010: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, 10 - 13 August 2010.

L. Muller. An adaptive large neighborhood search algorithm for the multi-mode resource-constrained project scheduling problem. l, Department of Management Engineering, Technical University of Denmark Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark, 2010.

L. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, Volume 218(Issue 3):614–623, 2011.

K. Nurmi and J. Kyngas. A conversion scheme for turning a curriculum-based timetabling problem into a school timetabling problem. In *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*, 2008.

K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *The Journal of the Operational Research Society*, 54(3): 230–238, 2003.

N. Pillay. An overview of school timetabling research. In *Proceedings of the International Conference on the Theory and Practice of Automated Timetabling*, pages 321–335, Belfast, United Kingdom, 2010.

G. Post, S. Ahmadi, and F. Geertsema. Cyclic transfers in school timetabling. *OR Spectrum*, pages 1–22, 2010. ISSN 0171-6468. 10.1007/s00291-010-0227-y.

G. Post, J. H. Kingston, L. di Gaspero, B. McCollum, and A. Schaerf. The third international timetabling competition (itc2011).
http://www.utwente.nl/ctit/hstt/itc2011/, 2011.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993. ISSN 0377-2217.

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4):190–204, 2009. ISSN 1097-0037. doi: 10.1002/net.20332.

G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728 – 735, 2012. ISSN 0305-0548. doi: 10.1016/j.cor.2011.05.005.

S. Ropke. Parallel large neighborhood search - a software framework. In *MIC 2009, The VIII Metaheuristics International Conference*, 2009.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

M. Salazar-Aguilar, A. Langevin, and G. Laporte. An adaptive large neighborhood search heuristic for a snow plowing problem with synchronized routes. In J. Pahl, T. Reiners, and S. Voss, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 406–411. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-21526-1.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, pages 1–14, 2010. ISSN 0254-5330. 10.1007/s10479-010-0709-y.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821. 10.1023/A:1006576209967.

K. Schimmelpfeng and S. Helber. Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, 29:783–803, 2007. ISSN 0171-6468. URL `http://dx.doi.org/10.1007/s00291-006-0074-z`. 10.1007/s00291-006-0074-z.

G. Schmidt and T. Ströhlein. Timetable construction – an annotated bibliography. *The Computer Journal*, 23(4):307–316, 1980. doi: 10.1093/comjnl/23.4.307.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems, 1997.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

H. ten Eikelder and R. Willemen. Some complexity aspects of secondary school timetabling problems. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 18–27. Springer Berlin / Heidelberg, 2001.

F. A. Tillman and T. M. Cain. An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science*, 18(11):664 – 682, 1972. ISSN 00251909.

L. Wolsey. *Integer programming*. Wiley-Interscience publication, 1998.

M. Yoshikawa, K. Kaneko, T. Yamanouchi, and M. Watanabe. A constraint-based high school scheduling system. *IEEE Expert*, 11(1):63 –72, feb 1996. ISSN 0885-9000. doi: 10.1109/64.482960.

E. Özcan. Towards an xml-based standard for timetabling problems: Ttml. In G. Kendall, E. K. Burke, S. Petrovic, and M. Gendreau, editors, *Multidisciplinary Scheduling: Theory and Applications*, pages 163–185. Springer US, 2005. ISBN 978-0-387-27744-8.

A Conversion to the XHSTT format

The XHSTT format (Post et al. (2012a)) is an XML-based format for (High) School Timetabling problem instances. It was used for the International Timetabling Competition 2011 (Post et al. (2012b)). Currently, 38 non-artificial datasets from 11 different countries are available.

In this section the problem instances of Lectio will be modeled in the XHSTT format. This will allow us to easily publish our instances. However, some aspects of HSTTP cannot currently be modeled with XHSTT, which is discussed in Section A.2. It is assumed throughout this section that the reader has in-depth knowledge of XHSTT.

Times

- `TimeGroups`: One day-`TimeGroup` is created for each day $d \in \mathcal{D}$. Furthermore, if the instance is a two-week instance, a week-`TimeGroup` representing each week is also created. Furthermore a `TimeGroup` is created for each neighbor-day pair, as described in Section 2.2.3, which contain all times of the respective days.
- `Time`: One `Time` is created for each timeslot $t \in \mathcal{T}$.

Resources

- `ResourceTypes`: Three `ResourceTypes` exist, namely `Room`, `Student`, `Teacher`. These correspond analogously to the sets $\mathcal{R}$ and $\mathcal{A}$ (students and teachers).
- `Resources`: For each room $r \in \mathcal{R}$ and each entity $a \in \mathcal{A}$, a `Resource` of corresponding type is created.

Events

- `EventGroups`: For each class $c \in \mathcal{C}$, a corresponding `EventGroup` is created. The members of an `EventGroup` are the events where the class participates. Notice that this is very similar to the definition of `Courses`, but since an event can contain more than one class, we use `EventGroups` instead of `Courses`.
- `Events`: A conversion from HSTTP EventChains to XHSTT-`Events` is now described. Events are either combined into the same event or linked together using constraints (i.e. certain events should be placed in the same or immediately following timeslot as other events).
  - Denote the set of entities, the set of classes, and the set of eligible rooms for event e by $\mathcal{A}_e$, $\mathcal{C}_e$ and $\mathcal{R}_e$, respectively. If for EventChain ec there exists two events $e_1, e_2 \in ec$ for which the set of entities, classes, eligible rooms, or locked room are different, i.e. if $\mathcal{A}_{e_1} \neq \mathcal{A}_{e_2}$, $\mathcal{C}_{e_1} \neq \mathcal{C}_{e_2}$, $\mathcal{R}_{e_1} \neq \mathcal{R}_{e_2}$, or $LR_{e_1,r} \neq LR_{e_2,r}$, then all events in EventChain ec must be linked using constraints.
  - If any event $e \in ec$ should be placed alongside other events, i.e. $S_e \neq \emptyset$, then all events in EventChain ec must be linked using constraints.
  - If none of the above applies, events are combined into one `Event`.

  Figure 6 illustrates conversion of some EventChains to XHSTT `Events`.

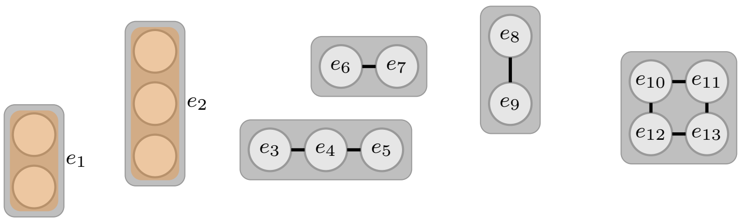


Fig. 6: *Conversion from EventChains to XHSTT events. e_1 and e_2 represent events which are combined into one XHSTT-`Event`. The remaining events are linked together using constraints.*

A.1 Constraints

In the following constraints of HSTTP is mapped to the XHSTT format. As in the MIP model (40), all constraints have `CostFunction` = `Sum`.

A.1.1 One timeslot - AssignTime

Only a single `AssignTime` constraint is needed, which applies to all events. `Weight` is set equal to 1, and `Required` equals `true`.

A.1.2 One room - AssignResource

A single `AssignResource` constraint is created, which applies to all events, and has `Role` = `Room` and `Required` = `true`. The `Weight` is set equal to 1.

A.1.3 Do not split events - SplitEvent

No events should be split, so all events are grouped by their duration, and for each group a single `SplitEvent` constraint is created, which applies to these `Events`, have `Required` = `true`, `Weight` = 1000, `MinimumDuration` and `MaximumDuration` set accordingly, and `MinimumAmount` = `MaximumAmount` = 1.

A.1.4 Teacher unavailable times - AvoidUnavailableTimes

The set of unavailable timeslots for a teacher is known (these partly defines parameter $D_{e,t}$). Group teachers by this set of timeslots, and create a `AvoidUnavailableTimes` constraint which applies to these teachers, and the respective set of timeslots. Further, `Required` = `true`, and `Weight` = 1.

Unavailable times for students are skipped as these are usually artificial in the sense that students are only marked as unavailable in certain timeslots by preference. I.e. for students it is preferred that late timeslots on each day are only used if necessary.

A.1.5 Do not split EventChains over days - PreferTimes

Neither an event or an EventChain can be assigned timeslots such that it spans over several days. For each event, identify its feasible timeslots by its EventChain. E.g. if an event has events which must be placed in contigious positions, then it cannot be assigned the last timeslot on a day.

Group events by their set of feasible timeslots. Create a `PreferTimes` constraint which apples to the appropriate events and times, has `Required` = `true` and `Weight` = 1.

A.1.6 Eligible rooms - PreferResource

Each event must be constrained such that it is only assigned its eligible rooms. Identify a set of `Resources` by parameter $K_{e,r}$, and create an `PreferResource` constraint with `Required` = `true`, `Weight` = 1000 and `Role` = `Room`. If several events have the same set of eligible rooms, these `PreferResource` constraints can be grouped. Notice that the priority of rooms as defined by eq. (42) is ignored.

A.1.7 Entity and Room conflicts - AvoidClashes

Only one `AvoidClashes` constraint is defined, which applies to all rooms, students and teachers. The constraint has `Required` set to `true` and `Weight` $= 1$. The XHSTT format does not currently allow us to restrict `AvoidClashes` constraint to only check for clashes in a subset of events, as was done in eq. ($4'$) and eq. ($5''$). Therefore instances might have inevitable violations of hard constraints.

A.1.8 Required days off - ClusterBusyTimes

Group teacher-entities by their number of required days off D, skipping those which require no days off. For each of these groups, generate a `ClusterBusyTimes` constraint which applies to these entities, with `Minimum` $= 0$, `Maximum` $= |\mathcal{D}| - D$, `Required` $=$ `true`, `Weight` $= 1$ and `TimeGroups` equal to the set of timegroups representing days.

A.1.9 Days occupied penalty - ClusterBusyTimes

Create a `ClusterBusyTimes` constraint which applies to all teacher-entities, with `Minimum` $=$ `Maximum` $= 0$, `Required` $=$ `false`, `Weight` as set by eq. (44), and `TimeGroups` equal to the set of timegroups representing days.

A.1.10 Days off penalty - ClusterBusyTimes

Create a `ClusterBusyTimes` constraint which applies to all student-entities, with `Minimum` $=$ `Maximum` $= |\mathcal{D}|$, `Required` $=$ `false`, `Weight` as set by eq. (45), and `TimeGroups` equal to the set of timegroups representing days.

A.1.11 Neighbor day conflicts - SpreadEvents

Define an `SpreadEvents` constraint which applies to all `EventGroups` representing classes, with `Weight` as set by eg. (47) and `Required` $=$ `false`. The `TimeGroups` section contains all `TimeGroups` which define a neighbor-day pair, and all entries have `Minimum` $= 0,$ `Maximum` $= 1$. Notice that all neighborday conflicts are penalized for all classes, contrary to eq. (40m).

A.1.12 Penalize idle slots - LimitIdleSlots

Two `LimitIdleSlots` constraints are created, which applies to all student-entities and all teacher-entities, respectively. The `Weight` is set as by eq. (43), and both constraints have `Required` $=$ `false`, `Minimum` $=$ `Maximum` $= 0$, and `TimeGroups` representing days.

A.1.13 Events in same timeslot - LinkEvents

Events which should be placed in the same timeslot as others can be specified using the `LinkEvent` constraint. For each set of these events, create a `LinkEvents` constraint which applies to these events, with `Required` $=$ `true`, `Weight` $= 1$, and one `EventGroup` which represents all events.

A.1.14 Events in contiguous timeslots - OrderEvents

For events which should be placed in contiguous timeslots ('followers'), an `OrderEvents` constraints is created with `Required` = `true` and `Weight` = 1000. The constraint applies to all pairs of events (e_1, e_2) where e_2 should follow immediately after e_1. All pairs have `MinSeparation` = `MaxSeparation` = 0.

A.1.15 Class day conflicts - SpreadEvent

A single `SpreadEvents` constraint is created, which applies to all `EventGroups` representing classes, with `Required` = `true` and `Weight` = 1. The `TimeGroups`-section is set equal to the set of timegroups representing days, with every entry having `Minimum` = 0 and `Maximum` = 1, Notice that it is not possible to constrain the events for which this constraints is applied, as was done in eq. (25). This may lead to hard constraint violations if classes are part of several events in the same EventChain, and these events cannot be combined into the same `Event`, as described in the `Event`-conversion scheme.

A.1.16 Daily workload - LimitBusyTimes

Group all teacher-entities by their maximum number of work-hours per day W_a. For each of these groups, create a `LimitBusy` constraint which applies to these teachers, with `Minimum` = 0,`Maximum` = W_a, `Required` = `true`, `Weight` = 1, and timegroups representing days.

A.1.17 More than one timeslot - LimitBusyTimes

Create a `LimitBusy` constraint which applies to all teachers, with `Minimum` = 2, `Maximum` = $|\mathcal{M}|$, `Weight` as set by eq. (48), `Required` = `false`, and timegroups representing days.

A.1.18 Week stability - LimitBusyTimes

The week stability constraint (40ab) cannot be modeled entirely as-is. Instead it is assumed that all events of class c is assigned a timeslot. This assumption seems fair in light of the applied AssignTime constraint. Let $d_c = \sum_e J_{e,c}$ be the number of events containing class c. Now group classes by d_c and create a `SpreadEvents` constraint which applies to all `EventGroups` representing these classes, have `Required` = `false` and applies to two timegroups, each representing a week, with `Minimum` = $\lfloor \frac{d_c}{2} \rfloor$ and `Maximum` = $\lceil \frac{d_c}{2} \rceil$. Thereby a class with 5 event must have a week-distribution of 2/3 or 3/2, and a class with 6 events must have the distribution 3/3.

A.2 Summary

The following aspect of HSTTP are not modeled with the XHSTT format:

- Penalty for rooms with low priority for events, as defined in eq. (42). This is a minor flaw, as few events will have second and third-priority rooms.

- Events can be assigned a room, but not a timeslot. This also clashes with a hard constraint of HSTTP, however, it does not pose a major problem as such events could be filtered out if we imagine solving the XHSTT instance in a practical setting.
- All neighbor-day conflicts are penalized. As the penalty given is small, this is a minor flaw.
- All room conflicts, entity conflicts and class day conflicts are penalized, which might give inevitable violation of hard constraints.
- Previous room and previous timeslot constraints (40aa) and (40z) are not added, even though it would be possible using PreferRoom and PreferTime constraints. However these constraints of HSTTP are more related to the practical scenery of Lectio, and therefore those does not seem well-suited for an XHSTT instance.
- Constraints room stability ((40u) and (40t)) and days off week stability ((40ac)) cannot currently be modeled with XHSTT. Both are small flaws, as these represent soft-constraints with a small weight.
- The combining of students as described in the beginning of Section 2.2 is not taken into account. This would be possible by introducing separate constraints for different groups of students.

Even with these inconclusive aspects of XHSTT with respect to HSTTP, we still believe the conversion of Lectio instances have significant contribution. All hard-constraints can be modeled more or less accurate. The soft-constraints which are left out are not of very significant character. Furthermore the resulting datasets are the first ones to use the `OrderEvents` constraint, and the first ones to span multiple weeks.

A.3 XHSTT Datasets

Currently, copyright issues have been settled with three schools, such that three datasets have been made available in the archive XHSTT-2013. We hope to be able to make more datasets publicly available soon.

Table 6 shows statistics for the datasets converted into the XHSTT format. The heuristic described in Sørensen et al. (2012) is applied to all instances 10 times, each with a timelimit of 240 seconds, and the best solution found is shown in the table. It is seen that all found solutions contain hard constraint violations. As previously described, it is expected that in the majority of cases, the optimal solution will contain some violation of hard constraints.

Table 6: *XHSTT datasets statistics.*

Dataset	Times	Teach.	Rooms	Classes	Stud.	Events	Total duration	Best sol.
FalkonG2012	50	91	63	313	278	1120	1120	(101,19464)
HasserG2012	50	100	69	423	521	1475	1475	(319,24312)
VejenG2009	60	46	53	189	163	928	928	(2,23275)

A complex model of high school timetabling is presented, which originates from the problem-setting in the timetabling software of the online high school ERP-system Lectio.
An Integer Programming formulation is described in detail and a two-stage decomposition is suggested. It is proven that both of these formulations are NP-hard.
An heuristic based on Adaptive Large Neighborhood Search is also applied. Using 100 real-life datasets, comprehensive computational results are provided showing that the ALNS heuristic outperforms the IP approaches.
The ALNS heuristic has been incorporated in Lectio and is currently available to almost 200 different high schools in Denmark.
Furthermore, a conversion of the datasets into the XHSTT format is described and some datasets are made publicly available.

ISBN 978-87-92706-98-0

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk