Integer programming for the generalized high school timetabling problem

Kristiansen, Simon; Sørensen, Matias; Stidsen, Thomas Riis

Published in:
Journal of Scheduling

Link to article, DOI:
10.1007/s10951-014-0405-x

Publication date:
2015

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Integer Programming for the Generalized (High) School Timetabling Problem

Simon Kristiansen · Matias Sørensen · Thomas R. Stidsen

Abstract Recently the XHSTT format for (High) School Timetabling was introduced, which provides a uniform way of modeling problem instances and corresponding solutions. The format supports a big variety of constraints, and currently 38 real-life instances from 11 different countries are available. Thereby the XHSTT format serves as a common ground for researchers within this area. This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a Mixed-Integer linear Programming (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results show that our approach is able to find previously unknown optimal solutions for 2 instances of XHSTT, and proves optimality of 4 known solutions. For the instances not solved to optimality, new non-trivial lower bounds were found in 11 cases, and new best-known solutions were found in 9 cases. Furthermore the approach is shown to be competitive with the finalist of Round 2 of the International Timetabling Competition 2011.

Keywords High School Timetabling · XHSTT · Integer Programming

1 Introduction

The problem of scheduling lectures to time slots and/or resources at high schools is known as the High School Timetabling (HST) problem. This is an important problem for high schools in many countries, and a large amount of different solution approaches have been proposed, see the survey Schaefer (1999).

It is well recognized that the specifications of the HST problem varies significantly depending on the country of which the problem originates, and that the problem in general is hard to solve. With the introduction of the XHSTT format (Post et al, 2012a), a large number of instances from various origins became publicly available in standardized form. The format is based on the Extensible Markup Language (XML) standard, and all instances are available online (Post, 2013a). One purpose of the format is to serve as a common test-bed for school timetabling, in an attempt to promote research within this area. In this context, "school timetabling" denotes the area covering high school timetabling and university course timetabling, as the format has also been shown capable of modeling some instances of the latter problem (see Kingston (2013a) for an overview of educational timetabling problems).

This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a Mixed-Integer linear Programming (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results are performed for all the real-life instances currently available. Thereby we are able to find previously unknown optimal solutions, and prove optimality of already known solutions.

To the best of our knowledge, all previous solution methods for the XHSTT format have been heuristic in nature. Therefore no proof of optimality has been made
for any instance, except for those instances where a solution with objective value 0 is known, since 0 is a trivial lower bound for any XHSTT instance. The obvious advantage of Integer Programming (IP) over heuristic methods is the capability to issue certificates of optimality. Therefore it is remarked that a big advance within general-purpose MIP solvers has happened in recent years, see e.g. Bixby (2012). Even though the MIP we will present is inevitable complex in nature, it will be shown that it can be used to find optimal solutions for several instances of the XHSTT archive ALL_INSTANCES. For those instances where an optimal solution cannot be found, we are able to show a non-trivial lower bound on optimum in the majority of cases. These are significant results for high school timetabling in general.

The outline of this paper is as follows. Section 2 presents related literature. Section 3 presents the MIP model of XHSTT. Section 4 describes computational results. Finally, Section 5 concludes and describes future research possibilities.

2 Related Literature

The Third International Timetabling Competition (ITC2011) considered the HST Problem, based on instances of the XHSTT format (Post et al, 2012b). Four teams were part of the final round: The overall winner (Team Goal) used Simulated Annealing and Iterated Local Search to perform local search around a generated initial solution (Fonseca et al, 2012). Participant from the University of Nottingham (Hy-STT) used a method based on Hyper-heuristics (Kheiri et al, 2012). Team Lectio used Adaptive Large Neighborhood Search (ALNS) (Sørensen et al, 2012). Romrös and Homberger (2012) (Team HFT) used an Evolutionary Algorithm. The results of the competition can be found at the official homepage of ITC2011 (Post (2013a)).

Pimmer and Raidl (2013) describe a ‘timeslot-filling’ heuristic for XHSTT, which iteratively fills selected timeslots with sets of events. Two state-of-the-art solutions were found for instances of the archive XHSTT-2012. Ter Braak (2012) presents a Hyper-heuristic and several other heuristics for the XHSTT.

Valouxis et al (2012) describe a two-phase approach based on MIP used to solve the Greek case of the HST problem. This includes two instances which are part of the XHSTT project, and which were both solved to optimality (solutions were found with an objective value of 0).

In terms of Integer Programming and HST problems not based on XHSTT, the following contribution are mentioned: Santos et al (2012) present a Column Generation approach for establishing bounds for a set of datasets originating from Brazil. Birgas et al (2009) present an approach for Greek datasets where the Shift Assignment Problem is solved first, and the timetable is constructed on the basis on these work-shifts for teachers. The paper of Sørensen and Skidsen (2013) describes a complex MIP of the Danish case of high school timetabling, and establishes computational results for 100 real-life instances. Avella et al (2007) present an algorithm based on Very Large-Scale Neighborhood search where the neighborhood is explored by a MIP, for Italian cases of high school timetabling.

3 Problem Description and a Mixed Integer Programming Formulation

In this section a brief description of the specifications of the XHSTT format is given, and a MIP model is formulated. The entire documentation of XHSTT is available at Post (2013a). We do not intend to describe all properties of the format, but only those necessary to formulate the MIP.

An instance of XHSTT consists of times (denoted \( T \) in the following), time groups (denoted \( T_G \)), resources (denoted \( R \)), events (denoted \( E \)), event groups (denoted \( E_G \)) and constraints (denoted \( C \)). An event \( e \in E \) has a duration \( D_e \in \mathbb{N} \), and a number of event resources which we each denote \( er \in e \). An event resource defines the requirement of the assignment of a resource to the event, and this resource can be specified to be pre-assigned. If the resource is not pre-assigned, a resource of proper type must be assigned. Furthermore an event resource \( er \) can undertake a specific role \( r_e \), which is used to link the event resource to certain constraints.

It is the job of any solver for XHSTT to decide how each event should be split into sub-events. A sub-event \( se \) is defined as a fragment of a specific event \( e \in E \), has a duration \( D_{se} \leq D_e \), and inherits the requirement of resources defined by the event, such that each sub-event has the exact same resource requirements as the event. Let \( SE \) denote the entire set of sub-events, and let \( se \in e \) specify that sub-event \( se \) is part of event \( e \). The total duration of all sub-events for event \( e \in E \) in a solution cannot exceed \( D_e \). In our model formulation we create the ‘full set’ of sub-events with different lengths, i.e. all possible combinations of sub-events for a given event can be handled. E.g. if an event has duration 4, the set of sub-events for this event has the respective lengths 1, 1, 1, 2, 2, 3 and 4. As a constraint it is then specified that the summed duration of the active sub-events in a solution must equals 4. A sub-event is active
if it is assigned a starting time or a non-preassigned resource. An active sub-event is analogous to the concept of solution events defined in the XHSTT documentation.

The times $T$ are ordered in chronological order, and we let $p(t)$ denote the index number of time $t$ in $T$. A time group $Tg$ defines a set of times, and we let $t \in Tg$ denote that time $t$ is part of time group $Tg$.

Each constraint $c \in C$ is of a specific type, and the set $C$ can contain several constraints of the same type. Each constraint applies to certain events, event groups or resources, and penalizes certain characteristics of the timetable for these entities.

The following notation shorthand is made: By the notions $c \in C$, $r \in R$, $e \in E$, we denote that constraint $c \in C$ applies to event $e \in E$, resource $r \in R$, and event group $e \in E_g$, respectively.

The set of resources and times are both extended with a dummy-index, denoted the dummy-resource $r_D$ and the dummy-time $t_D$, respectively. These are necessary to ensure feasibility as we create all combinations of sub-events for each event, and not all of these can be assigned a time or the required resources without the duration of the active sub-events exceeding the duration of the event. Thereby these dummy-elements in fact represent that an event resource is not assigned a resource, and that a sub-event is not assigned a starting time, respectively.

### 3.1 Objective Function

Each XHSTT constraint penalizes timetables with certain characteristics, which contributes to the objective function of the MIP. Each constraint $c \in C$ has a set of point-of-applications (indexed by $p \in p_c$). With each point-of-application is associated a set of deviations (indexed by $d \in p$), and each deviation has a non-negative cost associated with it. How this cost is calculated depends on the constraint type. The cost of a point-of-application is found on basis of the cost of the deviations, and is influenced by an indicator on the constraint whether the constraint is a hard or a soft constraints, the weight of the constraint ($\omega_c \in \mathbb{N}$) and an indication of which CostFunction to use. For each constraint $c \in C$ we let the variable $s_{c,p,d} \in \mathbb{N}$ be the penalty value of the deviation $d \in p$ of the point-of-application $p \in c$. The set of point-of-applications and deviations should be understood in an abstract context; E.g. depending on the type of the constraint, a point-of-application could be an event, a resource, etc., and likewise for the deviations.

The objective of a solution consists of a value for both the hard constraints (denoted hard cost) and a value for the soft constraints (denoted soft cost). Usually the objective value of a solution is written as (hard cost, soft cost). The hard cost always takes priority over the soft cost, i.e. solutions are first ranked on their hard cost, and secondly on the soft cost. How this type of objective function is handled in context of a MIP is described in Section 3.4.

The cost of a constraint $c \in C$ which contains slack variable $s_{c,p,d}$ is denoted $f(s_{c,p,d})$.

$$f(s_{c,p,d}) = \omega_c \cdot \text{CostFunction}(s_{c,p,d})$$  \hspace{1cm} (1)

Five different types of CostFunction are allowed. The most trivial one is $\text{Sum}$, which simply sums the penalty value of all deviations for all point-of-applications. In the following each CostFunction is formulated in linear terms. Let the variable $obj_c \in \mathbb{N}_0$ denote the value of the of the CostFunction of constraint $c \in C$.

- **Sum**: Sum the deviations.
  $$\text{obj}_c = \sum_{p \in c, d \in p} s_{c,p,d} \quad \forall c \in C$$  \hspace{1cm} (2)

- **SumSquare**: Sum the squares of the deviations.
  To cope with this non-linear cost function, the variable $s_{c,p,d,i} \in \{0,1\}$ is introduced, which takes value 1 if the deviation $d \in p$ of the point of application $p \in c$ of constraint $c \in C$ has the penalty $i \in I$, and 0 otherwise. The objective value is defined as follows:

  $$\text{obj}_c = \sum_{p \in c, d \in p, i \in I} i^2 \cdot s_{c,p,d,i} \quad \forall c \in C$$  \hspace{1cm} (3)

However we also need to make sure that only a single integer value is selected,

$$\sum_{i \in I} s_{c,p,d,i} = 1 \quad \forall c \in C, p \in c, d \in p$$  \hspace{1cm} (4)

The amount of elements in the set $I$ determines the maximum possible penalty for a deviation, and thereby influence the maximum possible penalty for a constraint. To maintain optimality of the model, it is therefore important that the size of $I$ is selected sufficiently large. This is elaborated in Section 4.

- **SquareSum**: Square the sum of deviations.
  The binary slack variable $u_{c,p,j}^{\text{square}} \in \{0,1\}$ is introduced, which takes value 1 if the point of application $p \in c$ of constraint $c \in C$ has the deviation $j \in J$, and 0 otherwise.

  $$\text{obj}_c = \sum_{p \in c, j \in J} j^2 \cdot u_{c,p,j}^{\text{square}} \quad \forall c \in C$$  \hspace{1cm} (5)

$$\sum_{j \in J} u_{c,p,j}^{\text{square}} = 1 \quad \forall c \in C, p \in c$$  \hspace{1cm} (6)

$$\sum_{d \in p} s_{c,p,d} = \sum_{j \in J} j \cdot u_{c,p,j}^{\text{square}} \quad \forall c \in C, p \in c$$  \hspace{1cm} (7)
Like the set $I$, the size of the set $J$ must be selected sufficiently large to maintain optimality, see Section 4.

- **SumStep**: This penalizes by the number of positive deviations, irrespective of their value.
  The binary variable $u_{c,p,d}^{\text{sumstep}} \in \{0, 1\}$ is introduced, which takes value 1 if $s_{c,p,d} > 0$ for constraint $c \in C$, point-of-application $p \in c$ and deviation $d \in p$, and 0 otherwise.

$$\text{obj}_c = \sum_{p \in c, d \in p} u_{c,p,d}^{\text{sumstep}} \forall c \in C$$  \hspace{1cm} (8)

$$M \cdot u_{c,p,d}^{\text{sumstep}} \geq s_{c,p,d} \forall c \in C, p \in c, d \in p$$  \hspace{1cm} (9)

where $M \in \mathbb{N}$ is some sufficiently large number.

- **StepSum**: This CostFunction penalizes by investigating whether the constraint contains at least one positive deviation. If this is not the case, the penalty is 0.
  The binary variable $u_{c}^{\text{stepsum}} \in \{0, 1\}$ is introduced, which takes value 1 if there exists at least one positive deviation for the constraint $c \in C$, and 0 otherwise.

$$\text{obj}_c = u_{c}^{\text{stepsum}} \forall c \in C$$  \hspace{1cm} (10)

$$M \cdot u_{c}^{\text{stepsum}} \geq s_{c,p,d} \forall c \in C, p \in c, d \in p$$  \hspace{1cm} (11)

where $M \in \mathbb{N}$ is some sufficiently large number.

### 3.2 Mixed-Integer Programming Formulation

In this section the variables and the constraints of the MIP are described. As a basis for our approach is the variable $x_{se,t,er,r} \in \{0, 1\}$, which takes value 1 if sub-event $se \in \mathcal{E}$ has been assigned time $t \in T$ as starting time and resource $r \in \mathcal{R}$ is assigned to event resource $er \in se$, and 0 otherwise. To simplify notation, and to reduce the amount of non-zeros in the MIP, three auxiliary variables are introduced which all ‘inherit’ their values directly from $x_{se,t,er,r}$. Let the binary variable $y_{se,t} \in \{0, 1\}$ take value 1 if sub-event $se \in \mathcal{E}$ has been assigned time $t \in T$ as starting time, and 0 otherwise. The variable $v_{t,r} \in \mathbb{N}_0$ denotes the number of times resource $r$ is used in time $t$ by any set of sub-events. Let variable $w_{se,er,r} \in \{0, 1\}$ take value 1 if sub-event $se \in \mathcal{E}$ is assigned resource $r \in \mathcal{R}$ for event resource $er$, and 0 otherwise.

#### 3.2.1 Base Constraints

Besides all the constraints described in the specifications of the XHSTT, some basic constraints are needed to ensure feasibility. First of all we need to make sure that a sub-event is assigned only one starting time and that the number of resource assigned is exactly the same as the number of event resources of the event.

$$\sum_{t \in T, r \in \mathcal{R}} x_{se,t,er,r} = 1 \forall se \in \mathcal{E}, er \in se$$  \hspace{1cm} (12)

The following constraints variable $y_{se,t}$, and together with (12) ensures that a sub-event is not spread across multiple times. We denote by $|er|_{se,e}$ the amount of event resources for event $e$ of sub-event $se$.

$$\sum_{er \in se, t \in \mathcal{R}} x_{se,t,er,r} = |er|_{se,e} \cdot y_{se,t} \forall se \in \mathcal{E}, t \in T$$  \hspace{1cm} (13)
The link to variable \( v_{s,r} \) is shown in eq. (15). For time \( t \in T \) and \( s \in SE \) is found the set of possible starting-times for \( s \) which will cause resource \( r \in R \) to be used in time \( t \in T \). Let the set \( T_{s,t}^{\text{start}} \subseteq T \) be the set of times which sub-event \( s \) lies in if it is assigned starting time \( t \), i.e.

\[
T_{s,t}^{\text{start}} = \{ t' \in T \setminus t_D \mid \rho(t) - D_{s} + 1 \leq \rho(t') \leq \rho(t) \}
\]

\[
\sum_{s \in SE, r \in \text{se}, t' \in T_{s,t}^{\text{start}}} x_{s,t',r} = v_{t,r} \forall t \in T \setminus t_D, r \in R
\]  

(15)

The link to variable \( w_{s,r,t} \) looks as follows:

\[
\sum_{t \in T} x_{s,t,r} = w_{s,r,t} \forall s \in SE, r \in SE, t \in R
\]  

(16)

A sub-event cannot be assigned a start time if there is not enough continuous times after the start time to fulfill the duration, ensured by the constraint:

\[
y_{s,t} = 0 \forall s \in SE, t \in T \setminus t_D, \rho(t) + D_{s} - 1 > |T|
\]

(17)

**Active Sub-events**

As we create all possible sub-events for a given event, only a subset of these should be active in the final solution. The binary variable \( u_{s,t} \in \{0,1\} \) takes value 1 if sub-event \( s \in SE \) is active and 0 otherwise. Recall that a sub-event is active if its assigned a starting time, or if is assigned at least one non-preassigned resource. Let the parameter \( PA_{r} \in \{0,1\} \) take value 1 if event resource \( er \) has a preassigned resource, and 0 otherwise. The following constraints are imposed.

\[
\sum_{r \in SE, t \in TD} w_{s,t,r} \leq u_{s,t} \forall s \in SE, \forall r \in SE, PA_{r} = 0
\]  

(18)

\[
\sum_{s \in SE} y_{s,t} \leq u_{s,t} \forall s \in SE
\]

(19)

\[
\sum_{s \in SE} y_{s,t} + \sum_{r \in SE, t \in TD} w_{s,t,r} \geq u_{s,t} \forall s \in SE, er \in SE, PA_{r} = 0
\]

(20)

Constraint (20) is necessary to ensure events are not set as active, even though they do not meet the required criteria.

The duration of active sub-events for a given event must be exactly the same as the total duration of the event (by definition of a valid XHSTT solution).

\[
\sum_{s \in SE} D_{s} \cdot u_{s,t} = D_{e} \forall e \in E
\]

(21)

A number of constraints require that the value of a deviation \( V \in N \) should be within an upper-limit \( \bar{B}_{c} \in N \) and a lower-limit \( \underline{B}_{c} \in N \). This means that the penalty is defined as the amount which the value of a deviation exceeds \( \underline{B}_{c} \) or falls short of \( \bar{B}_{c} \). To simplify notation for these cases, we introduce the function \( \mathcal{U}_{\underline{B}, \bar{B}} V \), which is defined as follows:

\[
s \geq \mathcal{U}_{\underline{B}, \bar{B}} V \Rightarrow \left\{
\begin{array}{ll}
s \geq V - \underline{B} & \\
\underline{B} \leq s - V &
\end{array}
\right.
\]

(22)

Thereby the slack-variable \( s \) is forced to take the actual value of the imposed penalty.

A resource is busy at some time if it attends at least one solution event at that time, and busy at some time group if it is busy at one or more times within times of that time group. Let variable \( q_{r,t} \in \{0,1\} \) take value 1 if resource \( r \in R \) is busy in time \( t \in T \), and 0 otherwise. Similarly, let the binary variable \( p_{r,tg} \in \{0,1\} \) take value 1 if resource \( r \in R \) is busy in time group \( tg \in TG \), and 0 otherwise. The values of the two variables are determined by the following constraints.

\[
|SE| \cdot q_{r,t} \geq v_{t,r} \forall r \in R, t \in T \setminus t_D
\]  

(23)

\[
q_{r,t} \geq v_{t,r} \forall r \in R, t \in T \setminus t_D
\]  

(24)

\[
p_{r,tg} \geq q_{r,tg} \forall r \in R, tg \in TG, t \in t_g
\]  

(25)

\[
p_{r,tg} \leq \sum_{t \in t_g} q_{r,t} \forall r \in R, tg \in TG
\]  

(26)

Constraints (23) and (25) establishes lower bounds for the variables \( q_{r,t} \) and \( p_{r,tg} \) i.e. ensures that these must take value 1 in case the resource is actually busy in the respective time/time group. Constraints (24) and (26) are necessary to ensure that in case the resource is in fact not busy in the respective time/time group, variables \( q_{r,t} \) and \( p_{r,tg} \) must take value 0.

In the following the constraint types of the XHSTT documentation are formulated one by one. Each constraint type is described in brief terms, and we refer to Kingston (2013b) for more details. The formulation of these constraints in terms of a Mixed-Integer Linear Programming model has not been published before. We let the 'pseudo-set' \( \mathcal{C} \subseteq \mathcal{C} \) denote constraints of a certain type depending on the context, for instance the set of all assign resource constraints. Furthermore we in the following make use of the general slack variable \( s_{e,p,d} \) and will for each type of constraint implicitly define a corresponding slack variable with the appropriate indices for point-of-applications and deviations.
3.2.2 Assign Resources

**Applies to:** Events

**Point-of-application:** Event-resource

An Assign Resource constraint penalizes event resources that are not assigned resources. Specifically, the deviation at one point of application (an event resource with the appropriate role) is the sum of the duration of the sub-events of the respective event which are not assigned a resource. The cost of this constraint is given by:

\[
D_e = \sum_{s \in \mathcal{E}} D_{se} \cdot w_{se, er, r} = s_{s,e,t}^{assign} \quad \forall c \in \mathcal{C}, e \in c, \quad r \in e, \quad role_{er} = role_e
\]  

(27)

3.2.3 Assign Time

**Applies to:** Events

**Point-of-application:** Events

The assign time constraint penalizes sub-events which are not assigned times. The deviation at one point of application is the total duration of those sub-events derived from a specific event that are not assigned a time.

\[
D_e = \sum_{t \in T \setminus T_o} D_{se, t} \cdot y_{se, t} = s_{s,e,t}^{assign_{time}} \quad \forall c \in \mathcal{C}, e \in c
\]  

(28)

3.2.4 Split Events

**Applies to:** Events

**Point-of-application:** Events

A Split Event constraint places limits on the number of sub-events which may be derived from a given event, and on their duration. Let the parameters \(\mathcal{E}_{split}^{amount} \in \mathbb{N}\) and \(\overline{\mathcal{E}}_{split}^{amount} \in \mathbb{N}\) denote the minimum and maximum amount of sub-events which is used for a given event, respectively. And let \(\mathcal{E}_{split}^{max} \in \mathbb{N}\) and \(\overline{\mathcal{E}}_{split}^{max} \in \mathbb{N}\) be the minimum and maximum duration a sub-event can have for a given event, respectively.

The cost of this constraint is given by the number of sub-events whose duration is less than \(\mathcal{E}_{split}^{max}\) or greater than \(\overline{\mathcal{E}}_{split}^{max}\), and the amount by which the number of sub-events fall short of \(\mathcal{E}_{split}^{amount}\) or exceed \(\overline{\mathcal{E}}_{split}^{amount}\). The following constraints are imposed:

\[
\sum_{s \in \mathcal{E}} D_{se} \cdot w_{se, er, r} = s_{s,e}^{splitevent} \quad \forall c \in \mathcal{C}, e \in c
\]  

(30)

\[
\mathcal{E}_{split}^{max} > D_{se} \lor \overline{\mathcal{E}}_{split}^{max} < D_{se}
\]

The full deviation for constraint \(c \in \mathcal{C}\) and event \(e \in c\) is given by \(s_{s,e}^{splitevent} + s_{s,e}^{spliteventamount}\).

3.2.5 Distribute Split Event

**Applies to:** Events

**Point-of-application:** Events

The Distribute Split Event constraints set limits on the number of sub-events which may be derived from an event. Let \(D_e \in \mathbb{N}\) be the duration of the sub-events for which this constraint applies, and let \(\mathcal{E}_{split}^{min}\) and \(\mathcal{E}_{split}^{max}\) be the minimum and maximum number of sub-events of duration \(D_e\) which may be derived from a given event.

\[
\sum_{s \in \mathcal{E}} D_{se} \cdot w_{se, er, r} = s_{s,e}^{splitevent} \quad \forall c \in \mathcal{C}, e \in c
\]  

(31)

3.2.6 Prefer Resources

**Applies to:** Events

**Point-of-application:** Event-resources

This constraint defines that an event resource has different preferences for certain resources. The deviation is calculated by taking all the solution resources derived from the event resource that are assigned a resource that is not one of the preferred resources, and summing the duration of the sub-events of those resources that lie in. Let \(r \in c\) denote a preferred resource.

\[
\sum_{s \in \mathcal{E}} D_{se} \cdot w_{se, er, r} = s_{s,e}^{prefer} \quad \forall c \in \mathcal{C}, e \in c, \quad er \in e, \quad PA_{er} = 0, \quad role_{er} = role_e
\]  

(32)

3.2.7 Prefer Times constraints

**Applies to:** Events

**Point-of-application:** Events

Like the Prefer Resources constraint, events might also have preferences for certain times. The deviation is calculated for each event by summing the duration of all sub-events which is assigned a time which is not one of the preferred time. The constraint has an optional duration-property, denoted \(D_e \in \mathbb{N}_0\). If this property is given, only sub-events of duration \(D_e\) are considered. Let \(t \in c\) denote a preferred time.

\[
\sum_{s \in \mathcal{E}} D_{se} \cdot y_{se, t} = s_{s,e}^{prefer} \quad \forall c \in \mathcal{C}, e \in c
\]  

(33)
3.2.8 Avoid Split Assignments

**Applies to:** Evenets

**Point-of-application:** Evenets

Each solution resource can only have one resource assigned. However, when an event is split into sub-events, each of its event resources is split into several solution resources, and a different resource may be assigned to each of these solution resources. This constraint penalizes the assignment of different resources to these solution resources. The constraint examines all the solution resources derived from those event resources, and calculates the number of distinct resources assigned to them, ignoring unassigned solution resources. The deviation is then given by the amount by which this number exceeds 1. Let variable \( k_{c, e, r} \in \{0, 1\} \) take value 1 if event \( e \) is assigned to resource \( r \) with respect to avoid split assignment constraint \( c \), and 0 otherwise.

\[
\sum_{e \in E, c \in C \setminus \{c, e\}, r \in R} w_{s_e, c, r} \leq k_{c, e, r} \quad \forall c \in \mathcal{C}, e \in \mathcal{E}, t \in T \setminus t_D
\]

3.2.9 Spread Events

**Applies to:** Eventgroups

**Point-of-application:** Eventgroups

The Spread Event constraint has a deviation for each time group \( t_g \in \mathcal{G} \). Let \( B_{c, t_g} \) and \( \overline{B}_{c, t_g} \) be the minimum and maximum number of sub-events of a given event which can be placed in time group \( t_g \) of constraint \( c \), respectively. The deviation for each time group is given by the amount of which the number of sub-events for the given event which fall short of \( B_{c, t_g} \) or exceed \( \overline{B}_{c, t_g} \).

\[
U_{B_{c, t_g}} \cdot \overline{B}_{c, t_g} \leq \sum_{t \in T \setminus t_g} y_{s_e, c, t} \leq B_{c, t_g} \quad \forall c \in \mathcal{C}, e \in \mathcal{E}, t_g \in T \setminus t_D
\]

3.2.10 Link Events

**Applies to:** Eventgroups

**Point-of-application:** Eventgroups

A Link Event constraint specifies that some events should be assigned the same times. For each event of a given event group we build the set of times that the sub-events derived from that event are running (not just starting times). The deviation is then the number of times that appear in at least one of these sets but not in all of them. Let variable \( o_{e, t} \in \{0, 1\} \) take value 1 if at least one sub-event of event \( e \in c \in \mathcal{C} \) is assigned to time \( t \in T \), and 0 otherwise. Let variable \( l_{e, t} \in \{0, 1\} \) take value 1 if at least one event of event group \( e \in c \) is assigned to time \( t \in T \), and 0 otherwise. Constraints (37) and (39) ensure that these variables take correct values. The slack of Link Events constraints is defined in (40). Constraint (38) is necessary to restrict \( o_{e, t} \) to take value 1 in cases where the event is in fact not assigned to the particular time, which would avoid the penalty given by constraint (40), if any.

\[
\sum_{t' \in T \setminus t_g} y_{s_e, t'} \leq o_{e, t} \quad \forall e \in \mathcal{E}, s_e \in e, t \in T \setminus t_D
\]

\[
\sum_{t' \in T \setminus t_g} y_{s_e, t'} \geq o_{e, t} \quad \forall e \in \mathcal{E}, t \in T \setminus t_D
\]

\[
l_{e, t} \leq a_{e, t} \quad \forall e \in \mathcal{E}, t \in T \setminus t_D
\]

\[
l_{e, t} = o_{e, t} \leq s_{e, e, t} \quad \forall e \in \mathcal{C}, e \in e, e \in e, t \in T \setminus t_D
\]
3.2.12 Avoid Clashes

**Applies to:** Resources

**Point-of-application:** Resources

These constraints specify that certain resources should have no clashes, i.e., they should not be assigned two or more events simultaneously. The constraint produces a set of deviations at each point of application (each resource). For each time a resource is assigned two or more solution resources, there is one deviation with a value equal to the number of solution resources minus one.

\[ v_{t,r} - 1 \geq s_{c,r,t} \quad \forall c \in \mathcal{C}, r \in c, t \neq t_D \] (44)

3.2.13 Avoid Unavailable Times

**Applies to:** Resources

**Point-of-application:** Resource

An Avoid Unavailable Times constraint specifies that certain resources are unavailable for all events at certain times. The deviation is the number of unavailable times during which the resource attends at least one solution event. \( t \in c \) denotes that \( t \) is an unavailable time for constraint \( c \in \mathcal{C} \).

\[ \sum_{t \in c} q_{t,r} = s_{c,r,t} \quad \forall c \in \mathcal{C}, r \in c \] (45)

3.2.14 Limit Idle Times

**Applies to:** Resources

**Point-of-application:** Resources

A resource is idle at some time \( t \in t_g \) wrt. time group \( t_g \) if it is not attending any sub-events at that time, but it is busy at some earlier time and at some later time in time group \( t_g \). The Limit Idle Times places limits on the number of idle times a resource may have. Let the variables \( h^1_{r,t_g} \in \mathbb{N} \) and \( h^2_{r,t_g} \in \mathbb{N} \) indicate the ordinal number of the first and the last time, respectively, where resource \( r \in \mathcal{R} \) is busy in time group \( t_g \). Let \( |t| \) denote the amount of times in time group \( t_g \). Let the variable \( h_{r,t_g} \in \mathbb{N} \) denote the number of idle times of resource \( r \in \mathcal{R} \) in time group \( t_g \in \mathcal{T}_G \).

\[ |t_g| - (|t_g| - \rho(t)) \cdot q_{t,r} \geq h^1_{r,t_g} \quad \forall r \in \mathcal{R}, t_g \in \mathcal{T}_G, t \in t_g \] (46)

\[ \rho(t) \cdot q_{t,r} \leq h^2_{r,t_g} \quad \forall r \in \mathcal{R}, t_g \in \mathcal{T}_G, t \in t_g \] (47)

\[ h^2_{r,t_g} = h^1_{r,t_g} + 1 - \sum_{t \in t_g} q_{t,r} \quad \forall r \in \mathcal{R}, t_g \in \mathcal{T}_G \] (48)

For each resource of the constraint the deviation is calculated as follows. Calculate the total amount of idle times for all times \( t_g \in c \), and find the amount which this summed value falls short of minimum \( B_e \in \mathbb{N} \) or exceeds maximum \( B_e \in \mathbb{N} \). The deviation is then given by the sum of these amounts.

\[ U_{\mathcal{C},r} \sum_{t_g \in c} h_{r,t_g} \leq s_{c,r,t} \quad \forall c \in \mathcal{C}, r \in c \] (49)

3.2.15 Cluster Busy Times

**Applies to:** Resources

**Point-of-application:** Resources

A Cluster Busy Times constraint limits the number of time groups during which a resource may be busy. The deviation is given by the amount of by which the number of given time groups during which the resource is busy falls short of minimum \( B_e \in \mathbb{N} \), or exceeds maximum \( B_e \in \mathbb{N} \). Let \( t_g \in c \) denote a time group which this constraint applies to.

\[ U_{\mathcal{C},r} \sum_{t_g \in c} p_{r,t_g} \leq s_{c,r,t} \quad \forall c \in \mathcal{C}, r \in c \] (50)

3.2.16 Limit Busy Times

**Applies to:** Resources

**Point-of-application:** Resources

The Limit Busy Times constraints places limits on the number of times a resource may be busy within some time groups. These constraints produces a set of deviation at each point of application, one for each given time group. The deviations are given by the amount by which the number of times of the given time group that the resource is busy falls short of minimum \( B_e \in \mathbb{N} \), or exceeds maximum \( B_e \in \mathbb{N} \).

\[ -|t_g| \cdot (1 - p_{r,t_g}) + U_{\mathcal{C},r} \sum_{t_g \in c} q_{t,r} \leq s_{c,r,t} \quad \forall c \in \mathcal{C}, r \in c, t_g \in c \] (51)

3.2.17 Limit Workload

**Applies to:** Resources

**Point-of-application:** Resources

A workload of a solution resource is given by \( W_{e,se,er} = \frac{p_{r,t_g} \cdot L_e}{C_r} \), where \( L_e \in \mathbb{N} \) is the workload of event resource \( e \). The value is a floating-point number. A Limit Workload Constraint places limits on the total workload of solutions resources that certain resources are assigned to. The deviation of this constraint is the amount by
which the total workload of the solution resources assigned to that resource falls short of \( \bar{B}_e \in \mathbb{N} \) or exceeds \( \bar{B}_e \in \mathbb{N} \), rounded up to the nearest integer.

\[
U_{\bar{B}_e} \sum_{e \in C \cup \mathcal{F}} W_{e,se,cr} \cdot x_{se,cr} \leq S_{e,r}^{\text{limitworkload}}
\quad \forall c \in C, r \in e \tag{52}
\]

3.3 Mixed-Integer Programming Model

Given the definitions of all constraint types of XHSTT, and their respective slack variables, the objective of the model can be stated as eq. (53), setting aside the fact that some constraints are hard-constraints and some are soft-constraints.

\[
z = f(\text{assignres}) + f(\text{assigntime}) + f(\text{assignamount}) + f(\text{splitamount})
+ f(\text{splitidle}) + f(\text{orderres}) + f(\text{ordertime})
+ f(\text{spreadevent}) + f(\text{spreadevent}) + f(\text{spreadevent})
+ f(\text{orderevents}) + f(\text{avoidclashes})
+ f(\text{avoidsplitas}) + f(\text{avoidsplitas}) + f(\text{clusterbusy})
+ f(\text{clusterbusy}) + f(\text{limitbusy}) + f(\text{limitworkload})
\tag{53}
\]

The full MIP would therefore consists of minimizing \( z \), subject to eqs. (12) to (52). However, we take a different approach, as described in the next section.

3.4 Solution Approach

Even though it would be natural to simply input the MIP to a generic solver, a different approach is taken, which takes advantage of the XHSTT objective function. In this approach, the model is solved in two steps, denoted Step 1 and Step 2 in the following.

By the definition of the XHSTT objective, hard constraints always take priority over soft constraints. Therefore the following approach is taken for solving the model: In Step 1, a MIP is build which only contains the hard constraints. This MIP is given as input to the MIP solver, which is ran until the given time limit is reached, or until the model is solved to optimality. The found objective value is the hard cost of the solution. In case the time limit is reached, all variables are fixed to their final value (i.e. the value they take in the best found solution), and all the soft constraints are added to identify the true cost of the found solution. In case the MIP is solved to optimality, Step 2 is performed: All soft constraints are added and the solution process is warm-started from its previous state, with the time limit set to what remains of the original time limit. Furthermore a constraint is added which ensures that the optimal value of the hard cost is kept. Let \( z^{\text{hard}} \)
denote the sum of all slack variables belonging to the hard constraints. The following constraint is added:

\[
z^{\text{hard}} = \text{hard cost}
\tag{54}
\]

Now this extended MIP model is solved. The cost of the obtained solution, minus the hard cost found in Step 1, is the value of the soft cost. Notice that the nature of this solution method resembles lexicographic multi-objective optimization.

This approach takes advantage of the capability of MIP's to issue certificates of optimality. By this we mean that focus is put on the hard constraints until a solution is found with the optimal hard cost, and then we switch focus and consider the entire problem instance. If a heuristic solution method was used the inevitable question would be: When has sufficient effort been put into minimizing the hard cost?

4 Computational Results

This section presents computational results of the developed exact method, and has two primary intentions:

- How does the MIP compete with the heuristics of the ITC2011 round 2? Thereby the potential of this MIP approach can be evaluated on fair terms with well-performing heuristics. This is the subject of Section 4.1.
- Are we able to improve the best-known solutions for some instances, or even solve them to optimality? See Section 4.2.

All tests were run on a machine with an Intel i7 CPU clocked at 2.80 GHz and 12GB of RAM, running Windows 8 64 bit. In all cases the commercial state-of-art MIP solver Gurobi 5.5.0 was used. Two distinct sets of XHSTT instances have been used, both obtained from the XHSTT website (Post, 2013a). All obtained solutions have been verified as being valid using the evaluator HSEval (Kingston, 2013c).

As described in Section 3.1, an XHSTT objective consists of both a hard cost, and a soft cost, usually denoted (hard cost, soft cost). In case a solution has a hard cost of value 0, the objective is simply written as the soft cost, as is usually done in context of the XHSTT format.

As discussed in Section 3.1, the size of the sets \( I \) and \( J \) must be selected sufficiently high. Notice further that if the size of these sets is selected high, it
can have a big impact on the amount of variables in the model. It would be possible to select these sizes based on the properties of the constraints having Cost-Function \texttt{SumSquare} or \texttt{SquareSum}, however this is a quite complex operation as it must be derived based on each constraint-type. Instead we have selected $|Z| = 10, |J| = 10$, such that the maximum possible penalty is $9^2 = 81$. This means that we cannot claim optimality for solutions with objective $> 81$. An easy fix for this issue is to simply perform a re-run of Gurobi if a solution is claimed as optimal, with the size of the sets set to a higher value. The same is applicable for lower bounds of value $> 81$. We consider this is an implementation detail, and it will be seen that in practice it has no impact of the obtained results.

4.1 International Timetabling Competition 2011

This section compares the exact method with the results obtained by the finalists in Round 2 of ITC2011. In this round the solver for each participating team was tested on 18 previously unknown instances from the archive XHSTT-ITC2011-hidden. The time limit for all instances was nominated to 1000 seconds, but the organizers provided a tool to benchmark machines to find the machine-dependent equivalent of this time limit. On our machine this amended to 772 seconds. The possibility to benchmark machines facilitates a fair comparison with the competitors of ITC2011, except for the fact that the rules of ITC2011 did not allow the use of commercial software, which conflicts with our use of Gurobi. The aim of this section is therefore to demonstrate the potential of MIP in the context of timetabling (which is often overlooked), and not to claim how our approach would have positioned itself in ITC2011.

In terms of solver parameters, default settings are used, except for the \texttt{pseudo-parameter MIPFocus} which is set to value 1, emphasizing that we are mainly interested in finding incumbent solutions. Gurobi was only allowed to use a single CPU thread, as specified in the rules of ITC2011.

The participants of ITC2011 round 2 ran their algorithm 10 times on each instance, to eliminate the stochastic impact on the results. Since we are interested in the average performance of each participant for comparison, the following processing of the results was performed: For each instance and each participant, calculate both the average hard cost and the average soft cost, and round both to nearest integer. These numbers then denote how this participant performed on this instance.

Table 2 shows the obtained results. The value of "Avg. Ranking" was calculated as follows. Each solution method was ranked 1 to 5 on each instance, 1 being the best, and the average of these ranks was taken. According to this measure, the exact method of this paper is competitive with the methods used at ITC2011. Notice in particular that the exact method performs well on the smaller instances, and is generally not as competitive on the larger instances. On three instances the exact method gave the best results.

4.2 Aiming at Optimality

In attempt to produce new (optimal) solutions, the XHSTT archive \texttt{ALL_INSTANCES} was used, which contains 38 non-artificial instances. According to the website, this archive "contains all latest versions of the contributed instances". For 10 of the instances, a solution with cost 0 is already known, which constitutes an optimal solution by the definition of XHSTT. Hence these instances are skipped in this test. Notice that \texttt{ALL_INSTANCES} contains instances which originally came from XHSTT-ITC2011-hidden, but due to bug-fixes in some of the instances, we consider them as two separate sets of instances (by bug-fixes we mean altering of certain constraints, such that objective values are incomparable). We refer to (Post, 2013a) for instance-statistics.

This test was performed with the following setup: Gurobi is allowed to use all CPU cores (which is 8 in our case), and the time-limit is set to 24 hours for each instance. As initial solution for each instance, the current best known solution is provided. Default parameter settings of Gurobi were used. Table 3 shows the obtained results. A gap between an incumbent solution $x$ and a lower bound $LB$ is calculated by $\frac{x - LB}{LB}$.

For each instance a solution with XHSTT objective ($H, S$) is found, as well as a lower bound ($\underline{H}, \underline{S}$). By the definition of our solution method, we only have a lower bound on the soft cost $\underline{S}$ if an optimal solution for the hard cost is known, i.e. $H = \underline{H}$. If a lower bound or an objective value is not found we write ".". Notice that even though we give the current best known solution as starting solution, Gurobi might still not find a solution for Step 1, usually in case the instance in question is of huge size. In Table 3, both the gap for the hard cost and the soft cost is shown (in case the required costs and lower bounds are available).

Table 3 shows that our method obtains better solutions for 8 instances, 4 instances was solved to optimality, proving optimality of 3 previously known solutions and finding 1 new optimal solution. Furthermore, 11 new non-trivial lower bounds and 7 new best solutions have been established for the instances which were not solved to optimality.
4.2.1 Alternative Formulation

The Limit Idle Times constraint is known to be difficult for solvers to handle (Dorneles et al (2012)). In our formulation, this constraint is formulated using Big-M notation (constraints (46) and (47)), which can provide bad LP-relaxation, which in turn might slow down the solution process. Furthermore this constraint is part of most instances (29 of 38 instances in the ALL_INSTANCES archive), so an alternate formulation is proposed. The alternate formulation uses variable $h_{r,tg,t} \in \{0, 1\}$ which takes value 1 if resource $r \in R$ has an idle time in time $t \in tg$ in time group $tg$, and 0 otherwise. Constraints (46), (47) and (48) are replaced by

$$q_{r,t} - q_{r,t} + q_{r,t'} - 1 \leq h_{r,tg,t} \quad \forall r \in R, tg \in TG, t, t', t'' \in tg, \quad \rho(t') < \rho(t) < \rho(t'')$$

This yields more rows in the MIP; for each time group $tg \in TG$ the amount of additional constraints is $|tg|^3$. Furthermore, there is a great possibility that the amount of variables increases due to the extra dimension on the $h$ variable. However, no Big-M notation is used.

Due to the possible big increase in the size of the model, this alternative formulation is only tested on the smaller instances from archive ALL_INSTANCES, skipping those instances in which the optimal solution was found in the previous test (Table 3). Since the goal is to achieve good solutions as possible, we restart the procedure from the best found solution of Table 3 and run it for additional 24 hours. This test-setup means that we cannot compare the performance of the two formulations. Table 4 shows the obtained results. The table shows that this formulation is capable of finding 2 new optimal solutions. For the instances not solved to optimality, 6 lower bounds were improved, and new best solutions were found for 6 instances.

5 Conclusion

This paper has shown the first exact method for (High) School Timetabling instances in the XHSTT format. A solution method which takes advantage of the structure of the objective function of XHSTT has been proposed. For the most recent version of the archive ALL_INSTANCES, we were able to produce 2 new optimal solutions and prove optimality of 4 previously known solutions. For 11 other instances, new non-trivial lower bounds were shown. For the instances not solved to optimality, we were able to improve the best known solution in 9 cases.

<table>
<thead>
<tr>
<th>Instance</th>
<th>GOAL</th>
<th>HySST</th>
<th>Lectio</th>
<th>HFT</th>
<th>Exact method</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrazilInstance2</td>
<td>(1, 62)</td>
<td>(1, 77)</td>
<td>38</td>
<td>(6, 190)</td>
<td>46</td>
</tr>
<tr>
<td>BrazilInstance3</td>
<td>124</td>
<td>118</td>
<td>152</td>
<td>(30, 283)</td>
<td>39</td>
</tr>
<tr>
<td>BrazilInstance4</td>
<td>(17, 98)</td>
<td>(4, 231)</td>
<td>(2, 199)</td>
<td>(67, 237)</td>
<td>(5, 286)</td>
</tr>
<tr>
<td>BrazilInstance6</td>
<td>(4, 227)</td>
<td>(3, 269)</td>
<td>230</td>
<td>(23, 300)</td>
<td>682</td>
</tr>
<tr>
<td>ElementarySchool</td>
<td>4</td>
<td>(1, 4)</td>
<td>3</td>
<td>(30, 73)</td>
<td>3</td>
</tr>
<tr>
<td>SecondarySchool</td>
<td>1</td>
<td>23</td>
<td>34</td>
<td>(31, 1628)</td>
<td>(1604, 3878)</td>
</tr>
<tr>
<td>Aigio</td>
<td>13</td>
<td>(2, 470)</td>
<td>1062</td>
<td>(50, 3165)</td>
<td>(1074, 3573)</td>
</tr>
<tr>
<td>Italy_Instance4</td>
<td>454</td>
<td>6926</td>
<td>651</td>
<td>(263, 6379)</td>
<td>17842</td>
</tr>
<tr>
<td>KosenInstance1</td>
<td>(59, 9864)</td>
<td>(1103, 14800)</td>
<td>(275, 7141)</td>
<td>(989, 39670)</td>
<td>(3626, 2620)</td>
</tr>
<tr>
<td>Kottenpark2003</td>
<td>90928</td>
<td>(1, 56462)</td>
<td>(50, 69773)</td>
<td>(209, 84115)</td>
<td>(8491, 6920)</td>
</tr>
<tr>
<td>Kottenpark2005A</td>
<td>(31, 32108)</td>
<td>(32, 30445)</td>
<td>(350, 91566)</td>
<td>(403, 46373)</td>
<td>(2567, 53)</td>
</tr>
<tr>
<td>Kottenpark2008</td>
<td>(13, 33111)</td>
<td>(141, 89350)</td>
<td>(209, 98663)</td>
<td>-</td>
<td>(14727, 5492)</td>
</tr>
<tr>
<td>Kottenpark2009</td>
<td>(28, 12032)</td>
<td>(38, 93634)</td>
<td>(128, 93634)</td>
<td>(345, 99999)</td>
<td>(17512, 140)</td>
</tr>
<tr>
<td>Woodlands2009</td>
<td>(2, 14)</td>
<td>(2, 70)</td>
<td>(1, 107)</td>
<td>(62, 338)</td>
<td>(1801, 705)</td>
</tr>
<tr>
<td>Spanish school</td>
<td>894</td>
<td>1668</td>
<td>2720</td>
<td>(65, 13653)</td>
<td>(1454, 11020)</td>
</tr>
<tr>
<td>WesternGreece3</td>
<td>6</td>
<td>11</td>
<td>(30, 2)</td>
<td>(15, 190)</td>
<td>25</td>
</tr>
<tr>
<td>WesternGreece4</td>
<td>7</td>
<td>21</td>
<td>(36, 95)</td>
<td>(237, 281)</td>
<td>81</td>
</tr>
<tr>
<td>WesternGreece5</td>
<td>0</td>
<td>4</td>
<td>(4, 19)</td>
<td>(11, 158)</td>
<td>15</td>
</tr>
<tr>
<td>Avg. Ranking</td>
<td>1.72</td>
<td>2.67</td>
<td>2.50</td>
<td>4.44</td>
<td>3.61</td>
</tr>
</tbody>
</table>
Table 3 Performance of the MIP on ALL_INSTANCES. For each instance is listed the best previously known solution “Best”, and for the solution found by our approach is listed the time used to solve Step 1 “Time1”, the time used to solve Step 2 “Time2”. “Time” indicates the total solving time. All times have seconds as unit. Furthermore the objective “Obj” and the lower bound “LB” is listed. The percentage gap between the objective and the lower bound is divided into the gap for the hard constraints “Gap1” and the gap for the soft constraints, “Gap2”. Objectives in bold denote new best solution while optimal solutions are marked with “∗”.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Best</th>
<th>Time1</th>
<th>Time2</th>
<th>Time</th>
<th>Obj</th>
<th>LB</th>
<th>Gap1</th>
<th>Gap2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU BGS98</td>
<td>(3, 494)</td>
<td>&gt;86400</td>
<td>-</td>
<td>&gt;86400</td>
<td>(3, 494)</td>
<td>(-,-)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AU SAH96</td>
<td>(8, 52)</td>
<td>&gt;86400</td>
<td>-</td>
<td>&gt;86400</td>
<td>(8, 52)</td>
<td>(-,-)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AU TES99</td>
<td>(1, 140)</td>
<td>&gt;86400</td>
<td>-</td>
<td>&gt;86400</td>
<td>(1, 140)</td>
<td>(0,-)</td>
<td>100.0</td>
<td>-</td>
</tr>
<tr>
<td>BR Instance1</td>
<td>42</td>
<td>0</td>
<td>&gt;86400</td>
<td>&gt;86400</td>
<td>40</td>
<td>28.0</td>
<td>0.0</td>
<td>30.0</td>
</tr>
<tr>
<td>BR Instance2</td>
<td>5</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>5</td>
<td>1.0</td>
<td>0.0</td>
<td>80.0</td>
</tr>
<tr>
<td>BR Instance3</td>
<td>47</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>26</td>
<td>19.0</td>
<td>0.0</td>
<td>26.9</td>
</tr>
<tr>
<td>BR Instance4</td>
<td>78</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>61</td>
<td>42.0</td>
<td>0.0</td>
<td>31.2</td>
</tr>
<tr>
<td>BR Instance5</td>
<td>43</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>30</td>
<td>10.0</td>
<td>0.0</td>
<td>66.7</td>
</tr>
<tr>
<td>BR Instance6</td>
<td>60</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>60</td>
<td>14.0</td>
<td>0.0</td>
<td>76.7</td>
</tr>
<tr>
<td>BR Instance7</td>
<td>122</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>122</td>
<td>22.0</td>
<td>0.0</td>
<td>82.0</td>
</tr>
<tr>
<td>DK Falkoner2012</td>
<td>(2, 23705)</td>
<td>&gt;86400</td>
<td>-</td>
<td>&gt;86400</td>
<td>(2, 23705)</td>
<td>(0,-)</td>
<td>100.0</td>
<td>-</td>
</tr>
<tr>
<td>DK Hasseris2012</td>
<td>(293, 32111)</td>
<td>&gt;86400</td>
<td>-</td>
<td>&gt;86400</td>
<td>(293, 32111)</td>
<td>(-,-)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DK Vejen2009</td>
<td>(20, 18966)</td>
<td>&gt;86400</td>
<td>-</td>
<td>&gt;86400</td>
<td>(20, 18966)</td>
<td>(2,-)</td>
<td>90.0</td>
<td>-</td>
</tr>
<tr>
<td>UK StPoul</td>
<td>136</td>
<td>52</td>
<td>&gt;86348</td>
<td>&gt;86400</td>
<td>136</td>
<td>0.0</td>
<td>100.0</td>
<td>-</td>
</tr>
<tr>
<td>FI ElementarySchool</td>
<td>3</td>
<td>2</td>
<td>785</td>
<td>787</td>
<td>*3</td>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>FI HighSchool</td>
<td>1</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>1</td>
<td>0.0</td>
<td>100.0</td>
<td>-</td>
</tr>
<tr>
<td>FI SecondarySchool</td>
<td>88</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>88</td>
<td>77.0</td>
<td>0.0</td>
<td>12.5</td>
</tr>
<tr>
<td>GR UniInstance3</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>*5</td>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>GR UniInstance4</td>
<td>8</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>8</td>
<td>0.0</td>
<td>100.0</td>
<td>-</td>
</tr>
<tr>
<td>IT Instance1</td>
<td>12</td>
<td>1</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>12</td>
<td>12.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IT Instance4</td>
<td>78</td>
<td>12</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>62</td>
<td>27.0</td>
<td>0.0</td>
<td>56.5</td>
</tr>
<tr>
<td>XK Instance1</td>
<td>3</td>
<td>31</td>
<td>&gt;86399</td>
<td>&gt;86400</td>
<td>3</td>
<td>0.0</td>
<td>100.0</td>
<td>-</td>
</tr>
<tr>
<td>NL GEPRO</td>
<td>(1, 566)</td>
<td>&gt;86400</td>
<td>-</td>
<td>&gt;86400</td>
<td>(1, 566)</td>
<td>(0,-)</td>
<td>100.0</td>
<td>-</td>
</tr>
<tr>
<td>NL Kottenpark2003</td>
<td>1410</td>
<td>57</td>
<td>&gt;86343</td>
<td>&gt;86400</td>
<td>1410</td>
<td>(0,-)</td>
<td>0.0</td>
<td>-</td>
</tr>
<tr>
<td>NL Kottenpark2005</td>
<td>1078</td>
<td>88</td>
<td>&gt;86312</td>
<td>&gt;86400</td>
<td>1078</td>
<td>9.0</td>
<td>99.2</td>
<td>-</td>
</tr>
<tr>
<td>NL Kottenpark2009</td>
<td>9250</td>
<td>92</td>
<td>&gt;86308</td>
<td>&gt;86400</td>
<td>9035</td>
<td>160.0</td>
<td>0.0</td>
<td>98.2</td>
</tr>
<tr>
<td>ZA Woodlands2009</td>
<td>2</td>
<td>22</td>
<td>77878</td>
<td>77900</td>
<td>*0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>ES School</td>
<td>(3, 5066)</td>
<td>6525</td>
<td>&gt;79875</td>
<td>&gt;86400</td>
<td>357</td>
<td>322</td>
<td>0.0</td>
<td>9.8</td>
</tr>
</tbody>
</table>

Establishing optimal solutions and lower bounds is indeed a step forward for research within high school timetabling, and for the XHSTT format in particular. This gives researchers a possibility to compare their obtained solutions with an (optimal) lower bound, which is valuable for evaluating the quality of solutions.

As subjects for future research the following are mentioned. The MIP could be used in context of Two-Stage Decomposition (TSD), by first assigning times to events, and secondly assigning resources to event resources. Thereby the resource-assignments are done subject to the times assigned to events. Such an approach was used with great success in the paper of Lach and Lübbecke (2012) for the Curriculum-based University Timetabling Problem (the optimization problem used in the International Timetabling Competition 2007), and by Sørensen and Dahms (2013) for the real-world case of High School Timetabling in Denmark. In both of these papers, the TSD is theoretically capable of producing near-optimal results, even

1 Shorthand for instance FalkonergaardenGymnasium2012
2 Shorthand for instance HasserisGymnasium2012
3 Shorthand for instance VejenGymnasium2009
4 Shorthand for instance WesternGreeceUniversityInstance3
5 Shorthand for instance WesternGreeceUniversityInstance4
6 Kosovo.
through the problem is split into two separate MIPs. However, the XHSTT case is possibly less suited for this type of decomposition as instances might contain a majority of constraints related to resource assignments. Since the assignments to times for events are performed in the first stage of the decomposition, and because these assignments cannot be altered when the resource-assignments are performed, a TSD approach would possibly be heuristic in nature. Obviously, if an XHSTT instance have all resources preassigned to event resources, a TSD would be unnecessary.

Our MIP formulation is exponential in size by the amount of sub-events in the instance, as all possible combinations of sub-events are generated. A better formulation would be less dependent on this amount. One could for instance solve the model iteratively, and 'inject' new sub-events in the model on-the-fly. Another possibility would be to consider a formulation which simulate sub-events by an integer variable which define the lengths of each respective active sub-event. Such improved formulations are subject for future research.

### References


tice and Theory of Automated Timetabling (PATAT 2012), SINTEF, pp 489–492