A Column Generation Approach for Solving the Examination-Timetabling Problem

Gert Woumans , Liesje De Boeck , Jeroen Beliën , Stefan Creemers

PII: S0377-2217(16)00091-6
DOI: 10.1016/j.ejor.2016.01.046
Reference: EOR 13490

To appear in: *European Journal of Operational Research*

Received date: 24 May 2013
Revised date: 16 January 2016
Accepted date: 24 January 2016

Highlights

- We present a student-centric ETP with improve exam spread by dynamically allowing more versions of exams.
- We propose two column generation algorithms to solve exam-timetabling problems.
- Both models include dynamic spreading of exams by using spreading costs.
- Second model has post-processing: either by heuristic or integer program(IP).
- Model 2 performs marginally better than Model 1, but is less scalable.
- Student spreading costs for existing data sets were improved at the cost of more exam versions.

A Column Generation Approach for Solving the Examination-Timetabling Problem

Gert WOUMANS[a,c,1], Liesje DE BOECK[b,c], Jeroen BELIËN[b,c], Stefan CREEMERS[a,c]

[a] *IESEG School of Management (LEM-CNRS), Rue de la Digue 3, 59000 Lille, France Tel.: +33-3-20545892, Fax: +33-3-20574855*

[b] *KU Leuven campus Brussels, Center for Informatics, Modeling and Simulation, Warmoesberg 26, B-1000 Brussels, Belgium*

[c] *KU Leuven, Faculty of Business and Economics, Department of Decision Sciences and Information Management, Research Center for Operations Management, Naamsestraat 69, B-3000 Leuven, Belgium Tel.: +32-16-326958, Fax: +32-16-326624*

Abstract

In this article, we approach the Examination-Timetabling Problem (ETP) from a student-centric point of view. We allow for multiple versions of an exam to be scheduled to increase the spreading of exams for students. We propose two Column Generation (CG) algorithms. In the first approach, a column is defined as an exam schedule for every unique student group, and two Pricing Problems (PPs) are developed to generate these columns. The Master Program (MP) then selects an exam schedule for every unique student group. Instead of using branch-and-price, we heuristically select columns. In the second approach, a column consists of a mask schedule for every unique student group, and a PP is developed to generate the masks. The MP then selects the masks and schedules exams in the mask slots. We compare both models and perform a computational experiment. We solve the ETP at KU Leuven campus Brussels (Belgium) for the business engineering degree program and apply the models to two existing datasets from the literature.

Keywords: Timetabling, examination-timetabling problem, column generation, exam spreading

1 Introduction

Examination timetabling is a recurring and time-consuming task in educational institutions (Qu et al., 2008), and is defined as “the scheduling for the exams of a set of university courses, avoiding overlap of exams of courses having common students, and spreading the exams for the students as much as possible” (Schaerf, 1999). Examination timetabling has to deal with many requirements, which include: no clashes for students, a fair exam schedule with sufficient study time in between exams, preferences of lecturers, and availability of classrooms. Within those requirements, two types of constraints can be distinguished (Qu et al., 2008), namely soft and hard constraints. Some survey articles also define them as first-order and higher-order constraints (Schaerf, 1999) or primary and secondary constraints (Carter, 1986). Hard constraints (equivalent to first-order and primary constraints) are constraints that must be satisfied at all times. Soft constraints (equivalent to higher-order or secondary constraints) only need to be satisfied as much as possible. Qu et al. (2008) point out that the extent to which soft constraints are met, defines the quality of a timetable. This quality aspect is also the key difference between search and optimization problems as described by Schaerf (1999). Search problems only try to find a *feasible* solution by satisfying the hard constraints, while optimization problems try to find an

[1] Corresponding author
Email addresses: `g.woumans@ieseg.fr` (Gert Woumans), `liesje.deboeck@kuleuven.be` (Liesje De Boeck), `jeroen.belien@kuleuven.be` (Jeroen Beliën), `s.creemers@ieseg.fr` (Stefan Creemers)

optimized feasible solution. The main difference between the search and optimization problems is therefore the capability to incorporate soft constrains. This article presents two models that handle the latter type of Examination-Timetabling Problems (ETPs). However, contrary to the definition by Schaerf (1999) and the examination-timetabling literature in general, many universities schedule several versions of the same exam to further satisfy the soft constraints. Multiple versions of the same exam are created (1) to ensure that students do not have two exams at the same time, and (2) to improve the exam spread. Our models incorporate a trade-off between multiple versions of exams and the soft constraints such as spreading.

The many requirements imposed on ETPs impede manual timetabling. Automated timetabling has therefore been a highly researched topic in operational research (Qu et al., 2008). However, ILP models of ETPs cannot be solved with modern-day computers, as the number of variables in the LP model is too large. On top of that, examination timetabling has been proven to be NP-complete when introducing certain characteristics (Cooper and Kingston, 1996). For example, imposing constraints concerning the equal spread of exams, or dealing with the many different students who have shared courses increases the complexity of the problem immensely.

This article addresses the above issues from a more student-centric point of view. A trade-off function is used to create timetables for each student group separately, while trying to minimize extra exam versions being created. Existing methodologies schedule exams while trying to minimize the total spreading cost. We propose two algorithms that decompose the problem by using a Column Generation (CG) approach. Our first algorithm defines a column to be a complete examination timetable for every student group. The second algorithm defines a column to be a timetable containing examination time slots or mask slots without specifying the exams themselves. It constructs the final timetables by scheduling exams and taking the mask slots into account. The second model needs a post-processing step to construct the final timetables. The algorithms thus transfer a different part of the problem's complexity to the sub-problems. We compare both approaches in a computational experiment. We also apply both models to two existing datasets from the literature.

The remainder of this article is organized as follows: in the second section, we state our problem description and link this to the existing literature. We also give a brief overview of the techniques used to solve the ETP. The third section presents our first CG approach and the fourth section our second. We report the results in the fifth section. The last section concludes the article with the most important findings and with directions for future research.

2 Problem requirements and literature review

In this section, we present the ETP and the existing methodologies. We focus on the requirements defined in well-known problems and on existing literature related to the application of CG to ETPs.

2.1 The examination-timetabling problem

Educational institutions basically have two systems of offering classes to students. The first is the curriculum-based system in which the student follows a predefined curriculum for which the timetable has already been created (Demeester et al., 2012). The other system is a post-enrollment system, in which the timetables are produced after the students have selected their courses. In curriculum-based universities, it is easier to produce examination timetables because exams are only shared between several homogeneous groups. In post-enrollment universities, exams are shared between many different students who all have different curricula. European universities are typically a curriculum-based environment, but are incorporating more and more post-enrollment elements. Students follow a curriculum, but can make changes to their curriculum to widen their scope, to slow down or increase the pace, etc. In this setup, we aim to automate the examination timetabling, while optimizing exam spread for students and minimizing the number of times an exam is scheduled. For our problem setting, we define a *unique curriculum* as a *set of exams* for a *unique student group*. Exams in these sets can be shared across multiple unique curricula. If multiple students take the exact same set of exams, they form a unique student group. The group is indivisible, so exams are always scheduled for an entire student group. Multiple groups can take the exam at the same time, but this is not a requirement. Multiple versions of an exam are therefore possible. An exam version is an instance of an exam scheduled at a specific time slot. The number of versions is equal to the number of time slots for which the exam is planned.

The ETP is well known in the timetabling literature. Many articles in the proceedings of the PATAT (Practice and Theory on Automated Timetabling) conferences relate to this topic. Subsequently, a EURO (European

Association of Operational Research) Working group on Automated TimeTabling (WATT) was established (Burke et al., 1997). Qu et al. (2008) give an elaborate review of the problem and discuss several datasets: the *Toronto* dataset (*a*, *b*, *c*, *d*, and *e*), the *Nottingham* dataset (*a* and *b*), and the *Melbourne* dataset (*I* and *II*). An elaborate overview of the studies performed per type of problem is also provided. In 2007, PATAT and WATT issued an international competition specifying a problem now known as the ITC2007 problem (Queen's University of Belfast, 2007) where the goal function, soft constraints, hard constraints, and datasets were defined.

In this section, we discuss the requirements of our problem definition and link them to requirements of the existing problems in the literature.

i. Students should receive a fair exam schedule

The model should try to spread the exams as evenly as possible for every student group. Many other studies have incorporated this, ranging from a static approach such as not having two consecutive exams on the same day or overnight (Burke et al., 1998; Qu et al., 2008; Wijgers and Hoogeveen, 2007), to a more dynamic approach attributing spreading costs relative to the number of time slots between two exams. Problems *Toronto c*, *Toronto e*, and *Nottingham a* include a static spreading requirement in which students cannot have two (consecutive) exams on the same day. Problems *Toronto d*, *Nottingham b*, *Melbourne I*, and *Melbourne II* extend this and penalize consecutive exams on the same day and overnight. Laporte and Desroches (1984) introduced a dynamic spreading approach and utilized a cost parameter w_s for each pairwise combination of exams that are s time slots apart. These costs have been defined as $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, and $w_5 = 1$. The *Toronto b* problem evaluates the exam spread using the same dynamic spreading costs. Bullnheimer (1998) achieves dynamic spreading by maximizing study time instead of minimizing spreading costs. The ITC2007 problem both requires static and dynamic spreading. The total cost takes into account the number of times students have two subsequent exams as well as the number of times students have two or more exams within a predefined number of time slots. The *Toronto a* problem does not include any spreading measures. We employ a dynamic spreading cost as defined by Laporte and Desroches (1984). Our problem relates best to the *Toronto b* set, of which we will use two instances to test the performance of the models presented in this article.

ii. Exams should preferably be scheduled only once

To honor the first requirement, one could schedule a new version of an exam for every unique student group, but this is undesirable. Every version requires new exams to be drawn up by the lecturer and requires other practicalities such as the presence of lecturers at the examination of every version. Contrary to standard ETPs, most European universities schedule multiple versions of an exam. We did not encounter any previous research that incorporated this into the automated-timetabling algorithms. Surely, in current methodologies multiple versions of exams can be created for different student groups to ease the scheduling, but not dynamically. Our models optimize a trade-off between scheduling an exam more than once and maximizing exam spread. Since this trade-off is not present in the literature, our article does not relate to the existing datasets for this aspect.

iii. Total room capacity must not be violated

Given a finite number of rooms with a finite capacity, the total capacity must be respected. The exam capacity is equal to the maximum number of students that can take an exam at the same time slot. This is a very common constraint in the literature but not every study takes this into account. Only *Toronto e* and the *Nottingham* sets specify that the total room capacity should not be exceeded (Qu et al., 2008). Contrary to these aggregated capacities, the ITC2007 problem requires that exams are assigned to rooms and that individual room capacities must be respected. Our models do not schedule exams to individual rooms. We therefore use aggregated room capacities.

2.2 Existing methodologies

Extensive overviews of ETP-solving methodologies are available in several survey articles. Carter (1986) discusses the application of graph theory to ETPs, with a focus on heuristics that use graph-coloring. Schaerf (1999) defines the difference between school timetabling problems, University Course Timetabling Problems (UCTPs), and ETPs. The article also includes a list of previous survey articles. For ETPs, Schaerf (1999) reviews direct heuristics, graph-coloring, simulated annealing, genetic algorithms, and several other methods.

Burke and Petrovic (2002) contribute to the literature with an overview of these topics and also discuss constraint-based techniques, tabu search, hyper heuristics, and population-based techniques such as evolutionary algorithms and ant algorithms. Qu et al. (2008) update this effort. Many techniques have thus already been applied to the ETP.

Mixed Integer or Integer Programming (MIP or IP) formulations are straightforward to set up, but they are constrained in their operation by the size of the problem. Examples are an application at the university of Kuwait (Al-Yakoob et al., 2010) and a quadratic programming model by Bullnheimer (1998). Direct heuristics mostly mimic human action and improve an initial integer solution by altering one element at a time (Schaerf, 1999). Laporte and Desroches (1984) use a heuristic that first solves a basic MIP and then takes care of classroom requirements and the need to spread the exams. Dimopoulou and Miliotis (2001) use a heuristic algorithm to first schedule the exams and then improve the solution to satisfy other constraints in a similar manner to Laport and Desroches (1984). Recently, Arbaoui et al. (2015) present new preprocessing stages and an improved MIP model for the examination timetabling problem of ITC2007. The preprocessing stages exploit an exam-based conflict graph, together with hard constraints and room capacities, to reveal implicit conflict constraints between exams even when they have no students in common.

Graph-coloring formulations have been applied since 1967 (Qu et al., 2008). In these formulations, exams are represented by vertices and time slots by colors. Two nodes are connected by an edge when they share resources and cannot be scheduled for the same time slot. When coloring the graph, two adjacent vertices cannot receive the same color. Two exams that share, for example, the same students cannot be scheduled on the same day (Mehta, 1981). Carter (1986) focuses his review article on possible tactics concerning which node to color first. Wijgers and Hoogeveen (2007) use graph coloring to create "sub-exam schedules", a list of non-clashing exams for each time slot in their CG algorithm. Qu, Burke, and McCollum (2009) create a hyper-heuristic which selects the best method for coloring the nodes. The basic graph-coloring problem was also solved using CG by Mehrotra and Trick (1996), which makes it possible to solve large-scale basic ETPs where only a clash-free exam schedule has to be found. A recent example of a graph-coloring-based heuristic applied to an ETP, is the work of Kahar and Kendall (2010). They also compare the different requirements for the different ETP definitions. Abdul-Rahman et al. (2014b) assign a score for the difficulty of scheduling each examination using an adaptive linear combination of two graph-coloring heuristics and examinations are scheduled in an order based on this value.

Another category of solution methods consists of local search approaches. They are often combined with population-based techniques to search the solution space around initial feasible solutions. Popular local search techniques are simulated annealing (Zhang et al., 2010), great deluge (Kahar and Kendall, 2014), and tabu search (Kendall and Hussin, 2005; Pais and Amaral, 2011; White et al., 2004). Examples of population-based methods are genetic algorithms (Burke et al., 2010), memetic algorithms (Al-Betar et al., 2014), ant-colony algorithms (Dowsland and Thompson, 2004), particle swarm optimization (Marie-Sainte, 2015), and bee colony algorithms (Sabar et al., 2012; Alzaqebah and Abdullah, 2014; Alzaqebah and Abdullah, 2015). A comparison between different meta-heuristics applied to the ETP can be found in an article by Azimi (2004). The graph-coloring problem was also solved using local search approaches (Burke et al., 2010, 2007, 2005; Qu et al., 2009). Recently, Abdul-Rahman et al. (2014a) present a local search heuristic that decomposes the examinations into two sets: a set of difficult to schedule and a set of easy to schedule examinations. These sets are dynamically updated during the construction of a timetable to ensure earlier assignment of examinations that lead to infeasibilities in subsequent attempts. Moreover, the examinations within each set are ordered using different strategies based on graph coloring heuristics. Li et al. (2015) propose an evolutionary ruin and stochastic rebuild search algorithm in which optimization is achieved by an iterative process of component evaluation, solution disruption and stochastic constructive repair.

A last class of frequently used methods are hyper-heuristics, which combine different simpler heuristics. Instead of using only one heuristic, the problem at hand instructs the algorithm which (part of a) lower-level heuristic might be the most effective (Qu et al., 2009). Hyper-heuristics are increasingly utilized to generalize the method for solving ETPs (Demeester et al., 2012; Gogos et al., 2010; Malik et al., 2011; Sabar et al., 2011; Soghier and Qu, 2013, Burke et al. 2014, Qu et al., 2015).

In addition to the previous categories, some authors are also researching the possibility of parallelizing the optimization process of the ETP, to make greater use of distributed computing (Komar et al., 2011; Mansour and Sleiman-Haidar, 2011). Algorithms are designed to break up and work on different parts of the problem in parallel to speed up the optimization process.

Our models are based on CG. A Master Problem (MP) selects partial solutions (grouped decisions) that are grouped into one decision variable or a column. CG tries to solve large-scale Linear Programs (LPs) by only looking at a part of the possible columns and generating new columns that improve the goal function value. It decomposes the problem into a Restriced Master Problem (RMP) that constructs a solution, given available feasible columns, and a subproblem that generates new columns. Except for a technical report by Wijgers and Hoogeveen (2007), CG has not yet been applied to the ETP. CG can be defined as an algorithm that permits solving LP problems containing a huge number of variables of which many are non-basic. It deals with the complexity through decomposition and iterative improvement of a feasible solution. The technique has already been applied to other scheduling problems like the UCTP which is a problem similar to the classic ETP. For instance, CG has successfully been applied to a UCTP in Italy (Qualizza and Serafini, 2005). Wijgers and Hoogeveen (2007) first try to minimize the number of time slots using a CG approach. Their goal is to find a non-clashing schedule, indicating that no student has two exams at the same time. In their research, a column represents a schedule of non-clashing exams for one time slot. The model tries to minimize the number of columns (time slots) needed so that all exams are scheduled and students have a complete examination timetable. Note that this method does not assign a column to a *specific* time slot, as a column is generated for a random time slot. In the second part of their research, Wijgers and Hoogeveen (2007) are able to include availability of lecturers and to avoid a situation where students have two exams in one day. The columns (or schedules) are defined slightly differently. A column is now a schedule for one day and contains two time slots, so that the student day constraint is easy to incorporate. In order for lecturers to express their availability, columns are specific to a certain day. The model no longer minimizes the number of time slots, but it minimizes the artificial cost associated with choosing a schedule under the constraints mentioned above. The authors were unable to incorporate a dynamical spread of the exams. This inability is a direct result of the definition of a column. The spreading of exams is nevertheless very important for students. Our method therefore differs from the method of Wijgers and Hoogeveen. We transfer the complexity of the dynamic approach of minimizing spreading costs to the Pricing Problem (PP). A column is defined per unique student group, for which the spreading cost is calculated. The Master problem (MP) then selects a combination of columns that minimizes the weighted spreading and exam version costs, taking into account the room capacities. To avoid a situation in which some exams get an excessive number of exam versions (whereas others have only one version) we also assign a penalty to the maximum number of exam versions over all exams. In this way additional exam versions are spread over all exams.

3 Model 1: using a complete schedule as a column

In this section, we present the first model. We first define the columns that are used and we discuss the MP. Then, we cover the reduced cost of a column in this model and present a PP to generate these columns. We conclude this section with an overview of the algorithm that heuristically selects the columns in a process we call “fixing”.

3.1 Definition of a column

Our first model is the most intuitive. We use a complete schedule for a unique student group s as a column with index q. A column consists of a binary matrix $\boldsymbol{A_{sq}}$ where the columns represent the time slots and the rows represent the exams taken by student group s. The elements of the matrix, namely a_{sqet}, denote that exam e is scheduled at time slot t when the element equals 1, and 0 otherwise.

$$\begin{array}{llcccccccc} & \text{time slots:} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}$$

$$\text{exam } e \begin{cases} = a \\ b \end{cases} \quad \boldsymbol{A_{sq}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad (3.1)$$

The above example (3.1) shows a matrix for a fictional unique student group that has exams a and b within a period of eight time slots. In the example, a_{sqa5} and a_{sqb1} equal 1, indicating that the student group takes exam

a at time slot 5 and exam b at time slot 1. Each exam schedule column has a cost representing the fictional spreading cost. This cost c^S_{sq} is defined in the PP in section 3.4.

3.2 Master program

This model's MP will select the most beneficial combination of exam schedules, one for each unique student group. The MP trades off the cost of additional exam versions against the cost of a worse schedule for students, and treats the room capacity as a hard constraint. The MP is shown in (3.2)-(3.11) and uses the following definitions:

Sets and indices

E is the set of exams, index e

S is the set of student groups, index s

E_s is the subset of exams for unique student group s, $\forall s \in S$: $E_s \subseteq E$

S_e is the subset of student groups that take exam e, $\forall e \in E$: $S_e \subseteq S$, index s

T is the set of time slots, index t

Q_s is the set of schedules (columns) generated for student group s, index q

Parameters

a_{sqet}	=	binary, equals 1 if timetable q for student group s schedules exam e at time slot t
c^V	=	cost of scheduling an extra exam version
c^N	=	cost of the maximum additional exam versions over all exams
c^S_{sq}	=	cost of choosing schedule q for unique student group s, a function of study-time fairness
n^S_s	=	number of students in student group s
n^E_e	=	number of student groups taking exam e
r	=	total exam capacity of rooms

Decision variables

α_{sq}	=	binary, equals 1 if schedule q is chosen for student group s
ϵ^V_{et}	=	binary, equals 1 if exam e has a version scheduled at time slot t
ϵ^N_{et}	=	number of student groups in S_e not taking a scheduled version of exam e at time slot t
N	=	maximum number of additional exam versions

Dual variables

λ^T_{et}	=	dual prices for scheduling an exam e at time slot t due to version costs
λ^N_{et}	=	dual prices for not scheduling student groups when a version for exam e at time slot t is scheduled
λ^R_t	=	dual price for scheduling at time slot t, due to room capacities
λ^S_s	=	dual price for scheduling a timetable for student group s

$$\min z = c^N N + \sum_{e\in E}\sum_{t\in T} c^V \epsilon^N_{et} + \sum_{s\in S}\sum_{q\in Q_s} n^S_s c^S_{sq}\alpha_{sq} \tag{3.2}$$

s.t.

$$\sum_{s\in S}\sum_{q\in Q_s} a_{sqet}\alpha_{sq} - n^E_e\epsilon^V_{et} \le 0, \qquad \forall e \in E, \forall t \in T, \qquad \lambda^T_{et} \tag{3.3}$$

$$n^E_e\epsilon^V_{et} - \sum_{s\in S}\sum_{q\in Q_s} a_{sqet}\alpha_{sq} - \epsilon^N_{et} \le 0, \qquad \forall e \in E, \forall t \in T, \qquad \lambda^N_{et} \tag{3.4}$$

$$\sum_{s\in S}\sum_{q\in Q_s}\sum_{e\in E_s} a_{sqet} n^S_s \alpha_{sq} \le r, \qquad \forall t \in T, \qquad \lambda^R_t \tag{3.5}$$

$$\sum_{q\in Q_s} \alpha_{sq} = 1\,, \qquad \forall s \in S, \qquad \lambda^S_s \tag{3.6}$$

$$\sum_{t\in T} \epsilon^V_{et} - N \le 1, \qquad \forall e \in E \tag{3.7}$$

$$\alpha_{sq} \in \{0,1\}, \qquad \forall s \in S, \forall q \in Q_s \tag{3.8}$$

$$\epsilon_{et}^{V} \in \{0,1\}, \qquad \forall e \in E, \forall t \in T \tag{3.9}$$

$$\epsilon_{et}^{N} \geq 0, \qquad \forall e \in E, \forall t \in T \tag{3.10}$$

$$N \geq 0 \tag{3.11}$$

The goal function in equation (3.2) minimizes the total cost of exam versions and of the students' schedules. The first two terms are the costs for scheduling exams. During preliminary testing of this model, we encountered unbalanced outcomes. Some exams had up to six versions in a certain test case, while others had only one. We therefore added a variable N to the MP, which counts the maximum number of *additional* versions of any exam in the solution. The first term attributes a cost to N. The second term attributes a cost to the number of groups not taking an exam at the same time slot as the other student groups taking that exam. We use this as a proxy for extra exam version costs, as the absent student groups in one time slot will need at least one additional version of that exam at another time slot. We thus attribute the cost of an exam version to this. The last term incurs the total spreading costs for students. We weigh the cost of the student group exam schedules by the number of students in that group. The set of constraints in equation (3.3) forces a version to be scheduled at a time slot if at least one student group takes the exam at that time slot. We use a binary variable ϵ_{et}^{V} to assign a version of an exam when at least one unique student group has exam e at time slot t. A big-M constraint is used, because this aggregated formulation of master constraints reduces the tailing-off effect traditionally found in CG (Vanderbeck, 2005). This, however, does result in a less tight lower bound when relaxing the MP (Bosch and Trick, 2005). Instead of an arbitrary big value for M, we limit this effect by using the number of student groups taking that exam. In constraint (3.4), we then count the number of absent student groups for each exam version that is scheduled. Constraint (3.5) ensures that the total room capacity is not violated. For each student group, the MP will select exactly one exam schedule, which is ensured by (3.6). The dual prices used in the PP are listed next to each constraint. The integer constraints are defined in (3.8) and (3.10) for, respectively, the selection of columns and the scheduling of exam versions. We exclude scheduling exam versions on Saturday afternoons and on Sundays. Furthermore, a cost c^N is incurred to variable N in the goal function and we add (3.7) and (3.11). These additions make sure that N equals the maximum number of additional versions over all exams and that it is minimized in the goal function.

3.3 Reduced cost of a column

To obtain the dual prices, we relax the IP to an LP and replace (3.8) and (3.9) with (3.12) and (3.13). The relaxation enables the algorithm to calculate dual variables and a reduced cost k_{sq} of a column q for student group s. The cost for a new exam schedule is defined by c_{sq}^{S}. The reduced cost k_{sq} of column q for student group s, is given by (3.14) and is minimized in the PP to find new columns that can decrease the goal function value of the MP. A new column will only be added to the MP when its reduced cost is negative, i.e., when the beneficial impact outweighs the cost of the column).

$$\alpha_{sq} \in [0,1], \qquad \forall s \in S, \forall q \in Q_s \tag{3.12}$$

$$\epsilon_{et}^{V} \in [0,1], \qquad \forall e \in E, \forall t \in T \tag{3.13}$$

$$k_{sq} = c_{sq}^{S} - \sum_{t \in T} \sum_{e \in E_s} a_{sqet} (\lambda_{et}^{T} - \lambda_{et}^{N} + n_s^S \lambda_t^R) - \lambda_s^S \tag{3.14}$$

3.4 Pricing problem

Every iteration, the algorithm searches for a new column (exam schedule) for every unique student group that has the largest possible gain, i.e., the largest negative reduced cost). The columns make it possible to transfer some of the requirements to the PP, which results in a less complex MP. The transferred requirements are the two constraints of exam spreading for students. In this section, we model the PP as an IP to search for new columns by minimizing the reduced cost.

We use a cost definition as defined in (3.15), created by Laporte and Desroches (1984) and formalized by Qu, Burke, and McCollum (2009). Parameter C is a cost factor and I is the ideal number of time slots between two exams. The exponent in (3.15) measures the difference between the ideal number of time slots and the actual number of time slots between two exams. If a student has more time slots between two exams, the exponent will be negative, which results in a cost lower than 1. If a student group has fewer time slots between exams, the exponent is positive and a larger cost will be incurred. By defining a variable parameter I per student group, one can adapt this to the number of exams in that unique curriculum. However, to adhere to existing literature, we use the cost definition as defined in the article by Qu et al. (2009). The method defines costs with $C = 2$ and $I = 5$, for study times up to 5 time slots.

$$c_{t',t}^{spread} = C^{I-(t-t')} \tag{3.15}$$

We use equation (3.16) to calculate the impact of scheduling an exam e at time slot t on the reduced cost for student group s.

$$K_{et} = -\lambda_{et}^{T} - n_s^S \lambda_t^R \tag{3.16}$$

The PP that tries to find a new column for student group s is formulated in (3.17)-(3.23). The model uses the previous definitions with the following additions:

Parameters

K_{et}	=	impact on reduced cost for scheduling exam e at time slot t

Decision variables

δ_{tu}	=	binary, equals 1 if exams are scheduled at time slots t and $t+u$
κ_{et}	=	binary, equals 1 if exam e is scheduled at time slot t

$$\min\ z = \sum_{t\in T} \sum_{u\in 1..5: t+u \le |T|} c_u^{spread} \delta_{tu} + \sum_{e\in E_s} \sum_{t\in T} K_{et}\kappa_{et} \tag{3.17}$$

s.t.

$$\sum_{t\in T} \kappa_{et} = 1, \qquad \forall e \in E_s \tag{3.18}$$

$$\sum_{e\in E_s} \kappa_{et} \le 1, \qquad \forall t \in T \tag{3.19}$$

$$\sum_{e\in E_s} \left(\kappa_{et} + \kappa_{e(t+u)}\right) - 1 - \delta_{tu} \le 0, \qquad \forall t \in T, \forall u \in \{1, \dots, 5\}: t+u \le |T| \tag{3.20}$$

$$2\delta_{tu} - \sum_{e\in E_s} \left(\kappa_{et} + \kappa_{e(t+u)}\right) \le 0, \qquad \forall t \in T, \forall u \in \{1, \dots, 5\}: t+u \le |T| \tag{3.21}$$

$$\delta_{tu} \in \{0,1\}, \qquad \forall t \in T, u \in \{1, \dots, 5\}: t+u \le |T| \tag{3.22}$$

$$\kappa_{et} \in \{0,1\}, \qquad \forall e \in E_s, \forall t \in T \tag{3.23}$$

Goal function (3.17) minimizes the reduced cost. We use constraint (3.18) to make sure that every exam from the curriculum is scheduled exactly once and use constraint (3.19) to exclude these exams being scheduled at the same time slot. Constraint (3.20) takes into account the spreading costs when two exams are scheduled u time slots apart. Constraint (3.21) does the opposite and is a valid inequality to solve the MIP faster. We define the domain of the binary variables in (3.22) and (3.23).

The elements a_{sqet} of the columns' matrix $\boldsymbol{A_{sq}}$ are determined using (3.24). The cost c_{sq}^S of the column is determined by the spreading costs and is given in (3.25).

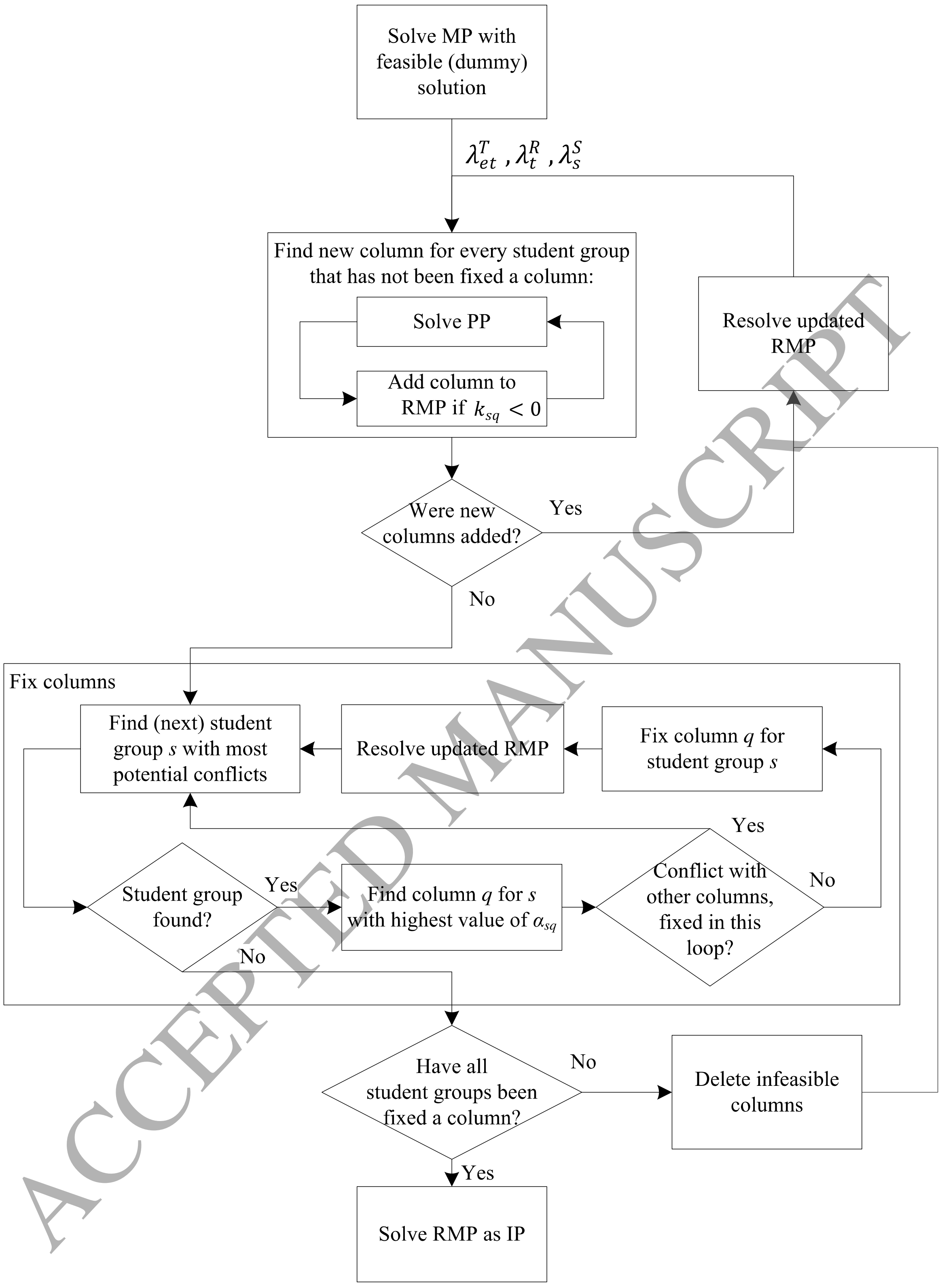
$$a_{sqet} = \kappa_{et} \tag{3.24}$$

$$c_{sq}^S = \sum_{t\in T} \sum_{u\in 1..5: t+u \le |T|} c_u^{spread} \delta_{tu} \tag{3.25}$$

Transferring the complexity of the spreading costs to the PP increases the flexibility to incorporate many requirements such as the spreading of exams. When testing the model, we found that the PP is still somewhat time-consuming on modern-day computers, demanding on average half of a second of computation time.

3.5 Algorithm for Model 1

The algorithm for the first model is based on a standard CG-scheme. The MP selects the individual exam schedules (columns) for student groups and schedules exam versions when the selected exam schedules require them. The dual prices from the relaxed MP are used in the PP to create a new exam schedule for a student group that is added to the MP. Figure 1 outlines the algorithm in a flowchart. CG must start with a feasible solution so that at least all hard constraints are satisfied. This can be achieved using methods such as the clique-partitioning method of Liu et al. (2011), which provides a feasible starting solution. Another possibility is to use a dummy solution that satisfies all constraints, but that is only selected at a big-M cost. We start by solving the relaxed RMP using a set of dummy columns that give us a first solution. This results in the necessary dual prices needed in the PP to search for new columns, or exam schedules. New columns that improve the RMP will be added to the RMP if the reduced cost k_{sq} of the column is strictly negative. If new columns are found, the RMP including these new columns is rerun, resulting in new dual prices. We then start the search for new columns again for every student group. This process is repeated until no columns are found and our relaxed RMP is fully solved.



Solve MP with feasible (dummy) solution
$\lambda_{et}^{T}, \lambda_{t}^{R}, \lambda_{s}^{S}$
Find new column for every student group that has not been fixed a column:
Solve PP
Add column to RMP if $k_{sq} < 0$
Resolve updated RMP
Were new columns added?
Yes
No
Fix columns
Find (next) student group s with most potential conflicts
Resolve updated RMP
Fix column q for student group s
Yes
Student group found?
Yes
Find column q for s with highest value of α_{sq}
Conflict with other columns, fixed in this loop?
No
No
Have all student groups been fixed a column?
No
Delete infeasible columns
Yes
Solve RMP as IP

Figure 1 Algorithm for solving Model 1

An optimal *relaxed* MP does not necessarily contain the columns needed in an optimal *integer* MP-solution. For this to be guaranteed, a time consuming branch-and-price algorithm is needed as discussed in Barnhart, Johnson, Nemhauser, Savelsbergh, and Vance (1998). Our algorithm heuristically selects a time slot at which a version for an exam is scheduled. The time slot of an exam that has the highest value of ϵ_{et}^{V} in the relaxed MP is fixed. New columns are then generated, while forcing exams to be scheduled at those time slots. Extra versions from existing columns are still allowed, but at an extra cost. Fixing a time slot for exams is done in batches, after which the new columns are generated. If a time slot is fixed for a specific exam, other exams that share students are excluded from being fixed at that time slot. Exams that share students with exams in the current batch of fixing, must also wait for the next batch so that new columns can be found for those student groups. In one batch of fixing examination time slots, time slots are fixed for as many as possible exams. The order of fixing for exams is determined by the number of potential conflicts with other exams. The exams with the highest number of potential conflicts are fixed first. We define a potential-exam-conflict matrix with dimensions $E \times E$. The elements $y_{e_1e_2}$ count the number of potential conflicts between the two exams by counting the number of shared student groups, and is defined in (3.26), where x_{es} equals 1 if student group s takes exam e, and 0 otherwise.

$$y_{e_1,e_2} = \sum_{s \in S} x_{es_1} x_{es_2} \tag{3.26}$$

This matrix is inspired by the conflict matrix proposed by Cole (1964), in which the elements c_{e_1,e_2} equal 1 if exam e_1 has students groups in common with exam e_2. Contrary to this binary logic, the elements in our matrix count the number of potential conflicts. We define the number of potential conflicts for an exam e_1 as Y_{e_1} in (3.27).

$$Y_{e_1} = \sum_{e_2 \in E \setminus \{e_1\}} y_{e_1e_2} \tag{3.27}$$

For exam e_1, we fix the time slot with the highest value of $\epsilon^V_{e_1t}$. We also remove the exam from our potential-exam-conflict matrix, since it has already been scheduled. Next, we reoptimize the RMP to take into account the newly fixed examination time slot.

This process is repeated until all exams have been fixed (and thus received a timetable). Finally, the RMP is solved as an IP to obtain the final values for ϵ_{et} and α_{sq} and the goal function value.

4 Model 2: using a mask per unique student group as a column

In this section, we discuss our second model. This model uses a mask with “mask slots” as a column. A mask slot is a time slot at which the student group is scheduled to take an exam from its curriculum.

4.1 Definition of a column

Column q for unique student group s is a vector $\boldsymbol{V_{sq}}$. Its binary elements v_{sqt} define the mask slots at time slots t for student group s. If v_{sqt} equals 1, student group s will take an exam at time slot t. In the example (4.1) below, student group s takes its three exams at the first, fourth, and eighth time slot. The number of assigned time slots is always equal to the number of exams in the curriculum E_s.

$$\begin{array}{ll} \text{time slots:} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix} \\ \boldsymbol{V_{sq}} = & \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{4.1}$$

Every column has a cost, which is equal to the spreading costs c^S_{sq} as discussed in section 3.2.

4.2 Master program

The MP assigns versions of exams to time slots and minimizes the number of versions by using masks as columns. We again select the most beneficial combination of columns in order to schedule the exam versions at the time slots when student groups are available. Explicitly assigning exam versions for student groups would result in a very large MP. We tested the idea of assigning exam e to student group s and time slot t with binary variable ϵ_{est}, which proved to be very time-consuming when imposing integrality constraints. Instead, we implicitly schedule exams for student groups. Exams are “overscheduled” when an exam version is scheduled, but the student group cannot take the exam at the scheduled time slot. Overscheduling can be the result of one of the following two reasons: either the student group does not have a mask slot scheduled at that time slot, or another exam version from that student group’s curriculum is already scheduled at that time slot. We minimize the number of times exams are overscheduled for a student group. For each time slot t, variable Y_{st}counts the number of overscheduled exams for the curriculum of student group s. As the overscheduling results in extra versions to be created at another time slot, we attribute a cost of scheduling extra exam versions. This has two effects. Firstly, this approach loses the ability to differentiate the cost of each version depending on the weight exam in a student's curriculum. Secondly, as the scheduling of exam versions for students is done implicitly, the goal function does not calculate the total costs in the same manner as the first model. The actual total cost is calculated in a post-processing step (see section 4.6). The MP minimizes the total cost that consists of the cost of overscheduling exam versions and the spreading costs for students. The MP is described in (4.2)–(4.11) and uses the following definitions:

Sets

E is the set of exams, index e

S is the set of student groups, index s

E_s is the set of exams for unique student group s, $\forall s \in S$: $E_s \subseteq E$

S_e is the set of student groups that take exam e, $\forall e \in E$: $S_e \subseteq S$, index s

T is the set of time slots, index t

Q_s is the set of masks (columns) generated for student group s, index q

Parameters

c^V	=	cost of scheduling an extra exam version
c^N	=	cost of the maximum additional exam versions over all exams
c^S_{sq}	=	cost of choosing mask schedule q for unique student group s, equals spreading costs for students
n^S_s	=	number of students in student group s
r	=	total exam capacity of rooms
v_{sqt}	=	binary, equals 1 if an exam can be scheduled for student group s at time slot t in mask q

Decision variables

α_{sq}	=	binary, equals 1 if schedule q is chosen for student group s
ϵ^V_{et}	=	binary, equals 1 if exam e has a version scheduled on time slot t
N	=	maximum number of additional exam versions
Υ_{st}	=	continuous variable, number of overscheduled exams for student group s at time slot t

Dual variables

λ^E_{st}	=	dual price for a mask slot for student group s at time slot t, due to overscheduling
λ^R_t	=	dual price for scheduling at mask slot t, due to room capacities
λ^S_s	=	dual price for scheduling a timetable for student group s

$$\min\ z = c^N N + \sum_{s\in S}\sum_{t\in T} c^V \Upsilon_{st} + \sum_{s\in S}\sum_{q\in Q_s} n_s c^S_{sq}\alpha_{sq} \tag{4.2}$$

s.t.

$$\sum_{t\in T}\epsilon^V_{et} \geq 1, \qquad \forall e \in E, \tag{4.3}$$

$$\sum_{e\in E_s}\epsilon^V_{et} - \sum_{q\in Q_s} v_{sqt}\alpha_{sq} - \Upsilon_{st} \leq 0, \qquad \forall s \in S, \forall t \in T, \qquad \lambda^E_{st} \tag{4.4}$$

$$\sum_{s\in S}\sum_{q\in Q_s} n^S_s v_{sqt}\alpha_{sq} \leq r, \qquad \forall t \in T, \qquad \lambda^R_t \tag{4.5}$$

$$\sum_{q\in Q_s}\alpha_{sq} = 1, \qquad \forall s \in S, \qquad \lambda^S_s \tag{4.6}$$

$$\sum_{t\in T}\epsilon^V_{et} - N \leq 1, \qquad \forall e \in E \tag{4.7}$$

$$\alpha_{sq} \in \{0,1\}, \qquad \forall s \in S, \forall q \in Q_s \tag{4.8}$$

$$\epsilon^V_{et} \in \{0,1\}, \qquad \forall e \in E, \forall t \in T \tag{4.9}$$

$$\Upsilon_{st} \in \mathbb{N}_0, \qquad \forall s \in S, \forall t \in T \tag{4.10}$$

$$N \geq 0 \tag{4.11}$$

The goal function (4.2) minimizes the total costs. As in Model 1, we weigh the spreading cost of the columns according to the number of students in the student group. Constraint (4.3) makes sure that every exam has at least one exam version. Decision variable Υ_{st} is calculated in constraint (4.4). We are unable to use aggregation on this constraint to counteract the tailing-off effect, as our dual prices would only offer information either per time slot or per student group to the PP. In (4.5) we include the room capacity constraint. Constraint (4.6) imposes that every student group is associated with exactly one mask schedule. Integrality definitions for mask selection variables and exam version variables are included in (4.8) and (4.10). During testing, we again found unbalanced solutions in exam versions as was the case in Model 1. We therefore added decision variable N, which counts the maximum number of additional versions over all exams in the solution. We add the cost of the maximum additional exam versions to the goal function and add constraint (4.7). We also define the domain of N in (4.11).

This formulation has some drawbacks. Because exam versions are not explicitly assigned to groups, there is no guarantee that a student group receives a complete exam schedule. In the students' mask slots, some exams from their set of exams may be scheduled multiple times whereas it is possible that others are not scheduled at all. Therefore, post-processing of the results is required.

4.3 Reduced cost of a column

To obtain the dual prices, the integrality constraints (4.8) and (4.10) are replaced with (4.12) and (4.13). Model 2 leads to the reduced cost k_{sq} for a new column q for student group s, specified in (4.14). The cost at which the new exam schedule for student group s is selected is defined by c^S_{sq}.

$$\alpha_{sq} \in [0,1], \qquad \forall s \in S, \forall q \in Q_s \tag{4.12}$$

$$\epsilon^V_{et} \in [0,1], \qquad \forall e \in E, \forall t \in T \tag{4.13}$$

$$k_{sq} = c^S_{sq} - \sum_{t\in T} (\lambda^E_{st} + n^S_s \lambda^R_t) v_{sqt} - \lambda^S_s \tag{4.14}$$

4.4 Pricing problem

We use a PP similar to the first PP proposed for Model 1 in section 3.2 to find a column with the largest negative reduced cost. The PP is now only concerned with the mask slots in which the exams are to be scheduled. The spreading costs are calculated as in (3.15). The impact of creating a mask slot at time slots t on the reduced cost is defined in (4.15).

$$K_t = -\lambda^S_{st} - n^S_s \lambda^R_t \tag{4.15}$$

The PP for student group s is formulated in (4.16)-(4.21). The model uses the previous definitions with in addition:

Parameters

K_t = impact on reduced cost for scheduling a mask slot at time slot t

Decision variables

δ_{tu} = binary, equals 1 if mask slots are scheduled at time slots t and $t + u$

κ_t = binary, equals 1 if a mask slot is scheduled at time slot t

$$\min z = \sum_{t\in T} \sum_{u\in 1..5: t+u\le |T|} c^{spread}_u \delta_{tu} + \sum_{t\in T} K_t \kappa_t \tag{4.16}$$

s.t.

$$\sum_{t\in T} \kappa_t = |E_s| \tag{4.17}$$

$$\kappa_t + \kappa_{t+u} - 1 - \delta_{tu} \ge 0, \qquad \forall t \in T, \forall u \in \{1..5\}: t + u \le |T| \tag{4.18}$$

$$2\delta_{tu} - \kappa_t - \kappa_{t+u} \le 0, \qquad \forall t \in T, \forall u \in \{1..5\}: t + u \le |T| \tag{4.19}$$

$$\delta_{tu} \in \{0,1\}, \qquad \forall t \in T, u \in \{1..5\}: t + u \le |T| \tag{4.20}$$

$\kappa_t \in \{0,1\}, \qquad \forall t \in T$ (4.21)

Goal function (4.16) minimizes the reduced cost. Constraint (4.17) enforces that the new column contains exactly as many mask slots as there are exams in the student group's curriculum. The program links the variables for calculating the spreading costs in (4.18). Constraint (4.19) is a valid inequality to speed up the solving process. In (4.20) and (4.21), we define the problem as a binary IP.

The elements v_{sqt} of the columns' matrix $\boldsymbol{V_{sq}}$ are determined using (4.22). The cost c_{sq}^{S} of the column is determined by the spreading costs and is given in (4.23).

$$v_{sqt} = \kappa_t \tag{4.22}$$

$$c_{sq}^{S} = \sum_{t \in T} \sum_{u \in 1..5: t+u \leq |T|} c_u^{spread} \delta_{tu} \tag{4.23}$$

4.5 Post-processing: heuristic

As discussed in section 4.2, Model 2 does not assign exam versions explicitly to student groups. The program does not give an exam schedule for each unique group of students, nor does it guarantee a complete schedule. Hence, it is necessary to revise the solution of the MP. We perform the post-processing using two approaches: a heuristic approach and a binary IP approach. Both approaches assign an exam version to a student group and start from the solution given by the integer formulation of the MP. This solution contains a mask schedule for every unique student group and the time slots for which the exams have an exam version.

To illustrate the heuristic approach, we use the following example. Student group 1 takes exams 1, 2, 3, and 4, and student group 2 takes exams 1, 2, and 3. Exam 1 has a version at time slot 1 ($\epsilon_{11}^{V} = 1$), exam 2 at time slot 3 ($\epsilon_{23}^{V} = 1$), exam 3 at time slots 5 and 7 ($\epsilon_{35}^{V} = \epsilon_{37}^{V} = 1$), and exam 4 at time slot 5 ($\epsilon_{45}^{V} = 1$). Student group 1 takes its exams at time slots 1, 3, 5, and 7 ($\alpha_{1q} = 1$ and $\boldsymbol{V_{1q}} = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1]$); student group 2 takes its exams at time slots 1, 5, and 7 ($\alpha_{2q} = 1$ and $\boldsymbol{V_{2q}} = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$).

In the heuristic, we use a matrix $\boldsymbol{X}$ in which the columns represent the time slots and the rows represent the unique student groups. The elements of the matrix x_{st} represent the exam student group s has at time slot t. The elements in curly brackets are not fixed and will be changed by the heuristic. The post-processing heuristic works as follows:

Step 1 For every element m_{st}, create a set containing all exams that are currently assigned to time slot t for group s. If no mask slot exists, the element is equal to the empty set. Otherwise, the element is a series of exams that have versions at that time slot. This is shown for the example below. Note that exam 2 does not have a version planned in a mask slot for student group 2. This is solved in step 3.

$$\boldsymbol{X} = \begin{bmatrix} \{1\} & 0 & \{2\} & 0 & \{3,4\} & 0 & \{3\} \\ \{1\} & 0 & 0 & 0 & \{3\} & 0 & \{3\} \end{bmatrix}$$

The following 3 steps are repeated until no more moves can be made.

Step 2 In each row, fix the exams for which only one version is planned in that row and drop the other exams from the series in the fixed element m_{st}. In our example, this results in the matrix below. If there are multiple exams in the same time slot with only one version scheduled, first fix the exams for which there already exists a version at that time slot; then fix the exam with the most versions in the same time slot for other student groups.

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0 & 2 & 0 & 4 & 0 & \{3\} \\ 1 & 0 & 0 & 0 & \{3\} & 0 & \{3\} \end{bmatrix}$$

Step 3 For each group s, add the exams in E_s that are not planned in a mask slot to the series that are not yet fixed. In our example, we must add exam 2 to the elements for student group 2.

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0 & 2 & 0 & 4 & 0 & \{3\} \\ 1 & 0 & 0 & 0 & \{3,2\} & 0 & \{3,2\} \end{bmatrix}$$

Step 4 In each element, fix the exams if a version already is fixed in that column. If multiple exams from an element are already fixed at that time slot, then choose the exam with the highest number of student groups already taking it. In our example, nothing needs to be done. Return to step 2.

Step 2bis We fix exam 3 in the last time slot for the first student group.

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0 & 2 & 0 & 4 & 0 & 3 \\ 1 & 0 & 0 & 0 & \{3,2\} & 0 & \{3,2\} \end{bmatrix}$$

Step 3bis No exams are missing.

Step 4bis Exam 3 exists at the last time slot, so we fix exam 3 in the second row at the last time slot and remove it from the other elements in that row.

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0 & 2 & 0 & 4 & 0 & 3 \\ 1 & 0 & 0 & 0 & \{2\} & 0 & 3 \end{bmatrix}$$

Step 2tris We fix exam 2 in the fifth time slot for the second student group.

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0 & 2 & 0 & 4 & 0 & 3 \\ 1 & 0 & 0 & 0 & 2 & 0 & 3 \end{bmatrix}$$

All exams are fixed, so the heuristic is stopped. The result of the heuristic is a matrix that contains a complete examination timetable for every student group.

4.6 Post-processing: integer model

Using the solution from Model 2, we can create an IP that assigns exams explicitly to student groups and time slots. The column selection variables α_{sq} resulting from the MP are now parameters. The IP used is described in (4.24)-(4.32), and has the same definitions as the MP in section 4.2 with the following alterations:

Parameters

α_{sq} = binary, equals 1 if schedule sq is chosen for student group s in the integer MP of Model 2

Decision variables

ϵ^P_{ets} = binary, equals 1 if exam e is scheduled at time slot t for unique student group s

$$\min\ z = c^N N + \sum_{e\in E}\sum_{t\in T} c^V_e \epsilon^V_{et} \tag{4.24}$$

s.t.

$$\sum_{t\in T} \epsilon^P_{ets} = 1, \qquad \forall s \in S, \forall e \in E_s \tag{4.25}$$

$$\epsilon^P_{ets} - \sum_{q\in Q_s} v_{sqt}\alpha_{sq} \le 0, \qquad \forall s \in S, \forall e \in E_s, \forall t \in T \tag{4.26}$$

$$\epsilon^P_{ets} - \epsilon^V_{et} \le 0, \qquad \forall s \in S, \forall e \in E_s, \forall t \in T \tag{4.27}$$

$$\sum_{e\in E_s} \epsilon^P_{ets} \le 1, \qquad \forall s \in S, \forall t \in T \tag{4.28}$$

$$\sum_{t\in T} \epsilon^V_{et} - N \le 1, \qquad \forall e \in E \tag{4.29}$$

$$\epsilon^V_{et} \in \{0,1\}, \qquad \forall e \in E, \forall t \in T \tag{4.30}$$

$$\epsilon^P_{ets} \in \{0,1\}, \qquad \forall e \in E_s, s \in S, \forall t \in T \tag{4.31}$$

$$N \ge 0 \tag{4.32}$$

Goal function (4.24) minimizes the totals cost of exam versions. We impose in constraint (4.25) that every exam a student group takes is scheduled. Constraint (4.26) ensures that an exam is assigned to a student group only if there is a mask slot for the student group. In (4.27), an exam version is created if an exam is scheduled for a student group at that time slot. Constraint (4.28) makes sure that only one exam is planned for a student group at a given time slot. Similar to the approach used in Model 1, we try to level the additional exam versions by means of (4.29). We define the problem as a binary IP in (4.30) and (4.31), and define the domain of N in (4.32).

In the IP post-processing approach, we only use the mask schedules to find a solution as opposed to the heuristic post-processing approach, which also uses the exam version data. If we were to impose the already scheduled versions in the integer MP, we would unnecessarily constrain the post-processing MIP.

4.7 Solution algorithm for Model 2

Figure 2 illustrates the algorithm for Model 2. The CG algorithm is equivalent to the first model, but with another fixing strategy and the addition of a post-processing step. Post-processing can be performed by using the heuristic or by using the IP model. During testing, we noticed strong tailing-off effects, especially for larger values of c^V and c^N. Since we were unable to counter this effect by aggregating constraints (as in Model 1), we chose to cut off the CG-process by using the Lagrangian bound as a lower bound on the optimal solution of the MP (Lübbecke and Desrosiers, 2005). The process is stopped if this bound is within 5% of the RMP's current goal function value.

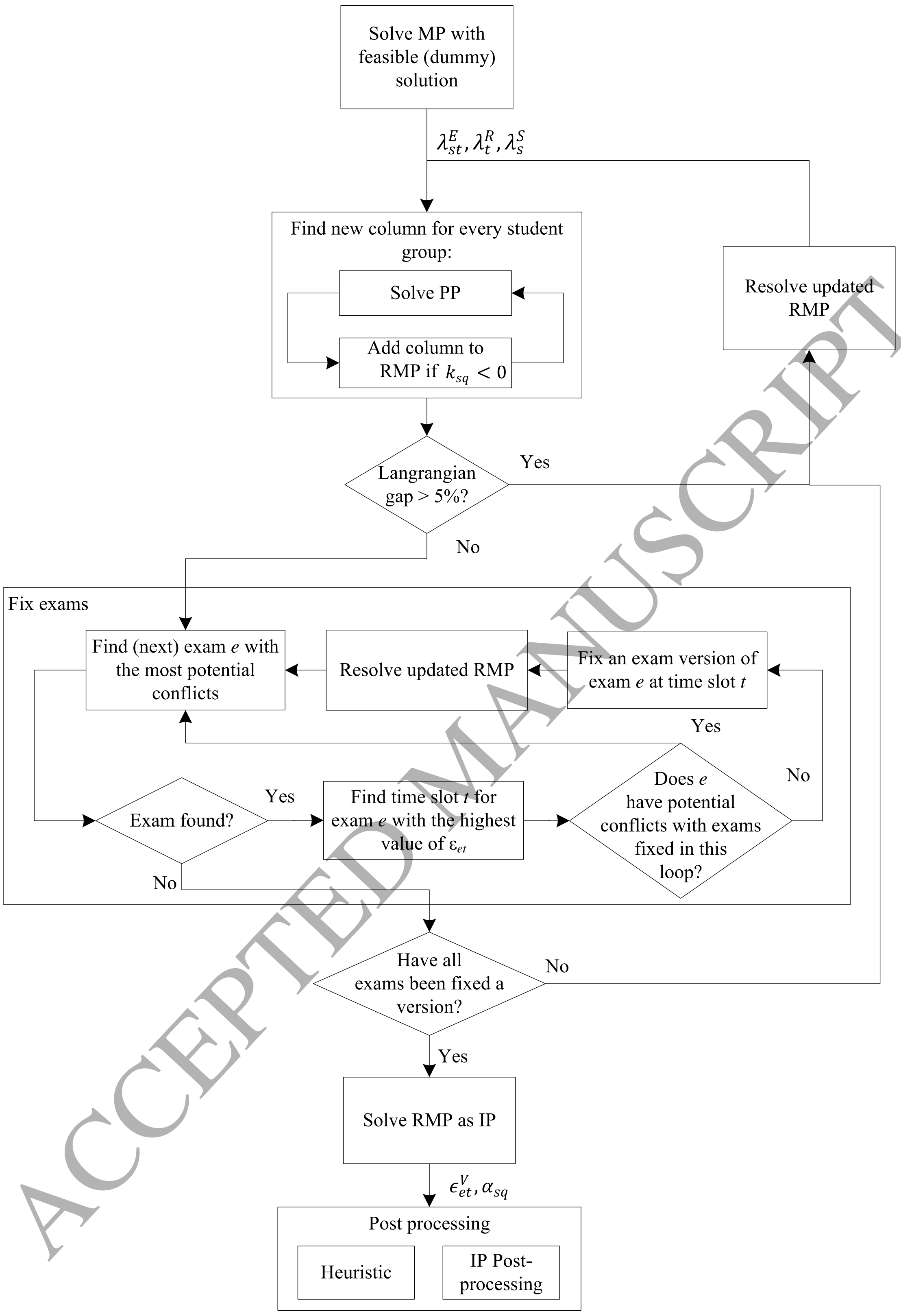

Solve MP with feasible (dummy) solution
$\lambda^E_{st}, \lambda^R_t, \lambda^S_s$
Find new column for every student group:
Solve PP
Add column to RMP if $k_{sq} < 0$
Resolve updated RMP
Langrangian gap > 5%?
Yes
No
Fix exams
Find (next) exam *e* with the most potential conflicts
Resolve updated RMP
Fix an exam version of exam *e* at time slot *t*
Yes
No
Exam found?
Yes
Find time slot *t* for exam *e* with the highest value of ε_{et}
Does *e* have potential conflicts with exams fixed in this loop?
No
Have all exams been fixed a version?
No
Yes
Solve RMP as IP
$\epsilon^V_{et}, \alpha_{sq}$
Post processing
Heuristic
IP Post-processing

Figure 2 Algorithm for solving Model 2

In this model, we cannot fix exams by fixing columns for student groups. We therefore fix exam versions in order of the number of potential conflicts. Exams with a higher number of potential conflicts are fixed first. We construct a potential-exam-conflict matrix, similar to the conflict matrix in Model 1, in which elements $z_{e_1e_2}$ count the number of potential conflicts between exams e_1 and e_2. Only exams that do not have potential conflicts with exams that were scheduled during this fixing loop, are allowed to be fixed. After the RMP has fixed a time slot for all exams, additional columns are generated for the last time. After this step, the RMP is solved as an IP. Next, because student groups are still assigned implicitly to exam versions, post-processing is required. During this step, we remove the fixing of all exam decision variables.

5 Results

In the following section, we test our algorithms on our own datasets containing one real-life dataset and three randomly generated sets. We also apply our algorithms to two datasets from the *Toronto* problem. We first present the test set-up. The results of our own datasets are then used to discuss and compare both approaches. Next, we present the results for the datasets from the literature. We conclude with a brief general discussion of the results.

5.1 Test set-up

The models are applied to a test case at the KU Leuven campus Brussels, Belgium. Data were collected for the bachelor and master degree of the Business Engineering program for the academic year 2011-2012. At that moment, 178 students were enrolled, which encompasses three bachelor years and two master years. We test our models using the data of the first semester of 2012. Next to this *HUB* dataset, we have also created three small random datasets *X1*, *X2*, and *X3*. These datasets have been created by randomly selecting 0 or 1 for each combination of 20 unique student groups and the preset number of exams. The number of students in each unique student group is also randomly selected. Finally, we compare our results with the best results reported in the literature for the *STA83* and *YOR83* instances from the *Toronto* dataset. By removing the total room capacity constraint, our models reduce to the *Toronto b* problem with multiple versions of exams. Table 1 shows a summary of relevant metrics of these datasets. We use the conflict density as a proxy of the complexity of the dataset, as proposed by Cole (1964). A higher ratio leads to a higher probability of arising conflicts when scheduling exams. We provide online access to our datasets at http://econ.kuleuven.be/gert.woumans/public.

Table 1
Descriptive statistics of the datasets

Dataset	*HUB*	*X1*	*X2*	*X3*	*STA83*	*YOR83*
Number of unique student groups	71	20	20	20	298	910
Number of students	178	59	118	129	611	941
Number of exams	39	24	22	35	139	181
Average number of exams per student group	5.69	4	3.9	5.22	8.7	6.42
Average number of students per exam	25.79	9.83	19.18	21.5	38.20	33.40
Conflict density	0.44	0.47	0.52	0.43	0.14	0.29
Total room capacity	120	40	80	90	-	-
Number of time slots	33	33	33	33	13	21

In our tests, we use a system with an AMD Phenom II 4X 965 CPU, clocked at 3.4GHz with 32GB RAM, running Windows 7 64-bit. We run IBM ILOG CPLEX Optimization Studio version 12.6 64-bit and use the OPL-modeling language to program the models and to script the algorithms.

5.2 Test results for the *HUB* datasets

We first present the results for our own test instances for different parameters of c^V and c^N. For completeness, in Table 2 we show all results for both models, i.e., the three different metrics by which we can assess the quality of an exam schedule: the average spreading cost for a student, the average number of exam versions per exam, and the maximum number of additional exam versions. We also report the total wall clock time. Figure 3 provides graphs of these results to illustrate how the models optimize for different parameters. Note that for the *HUB* dataset, the first model was also able to produce a timetable with no additional versions for any exam, if we keep c^V at 200 and increase c^N to 400.

Table 2
Results for the *HUB*, *X1*, *X2*, and *X3* datasets. The second model uses the IP post-processing technique.

Metric		**Average spreading cost per student**								**Average number of exam versions**								**Maximum number of extra versions (N)**								**Time (h:mm)**							
Dataset		***HUB***		***X1***		***X2***		***X3***		***HUB***		***X1***		***X2***		***X3***		***HUB***		***X1***		***X2***		***X3***		***HUB***		***X1***		***X2***		***X3***	
Model		**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**
c^V	c^N																																
5	**0**	5.95	8.04	0.81	0.61	0.94	2.97	2.36	1.89	2.67	2.05	1.83	1.33	1.77	1.05	1.80	1.26	7	3	3	1	2	1	4	2	0:28	1:58	0:14	0:13	0:05	0:14	0:16	0:31
5	**10**	5.62	7.83	0.97	0.63	0.88	2.58	2.02	1.87	3.18	2.18	1.67	1.25	1.73	1.14	2.14	1.31	5	3	3	1	2	2	3	1	0:28	1:54	0:11	0:13	0:05	0:13	0:18	0:31
5	**50**	5.52	7.65	0.78	0.56	1.31	2.07	2.07	1.92	2.72	2.10	2.00	1.29	1.82	1.09	2.26	1.26	4	2	2	1	1	1	2	1	0:31	1:32	0:12	0:13	0:04	0:08	0:19	0:17
5	**100**	5.77	8.03	0.92	0.71	1.22	2.94	2.16	2.02	3.00	2.10	1.54	1.29	1.86	1.14	2.23	1.31	4	2	2	2	1	2	2	1	0:34	1:46	0:11	0:12	0:05	0:11	0:19	0:17
5	**200**	6.78	7.58	0.86	0.61	1.35	3.17	2.20	1.98	2.46	2.08	1.88	1.29	1.73	1.14	2.14	1.31	3	2	2	2	1	1	2	1	0:31	2:11	0:12	0:13	0:05	0:08	0:19	0:13
10	**0**	7.09	12.03	1.97	0.93	1.14	1.71	2.29	2.30	2.05	1.59	1.54	1.21	1.59	1.18	1.83	1.17	5	5	2	1	2	1	4	2	0:37	1:30	0:08	0:13	0:04	0:05	0:14	0:28
10	**10**	7.05	8.93	1.22	0.69	1.30	4.05	2.71	2.09	2.46	1.62	1.54	1.29	1.55	1.00	1.46	1.26	4	2	3	1	2	0	2	1	0:39	1:32	0:08	0:11	0:05	0:03	0:12	0:27
10	**50**	7.07	8.85	1.76	0.68	1.51	3.63	2.78	2.17	2.08	1.56	1.54	1.25	1.82	1.14	1.63	1.23	3	2	2	1	1	1	2	1	0:38	1:27	0:08	0:09	0:05	0:09	0:13	0:18
10	**100**	7.17	8.55	1.32	0.92	1.61	3.92	2.56	2.19	1.85	1.51	1.75	1.25	1.45	1.05	1.77	1.14	2	2	2	1	1	1	2	1	0:39	1:39	0:08	0:13	0:06	0:04	0:13	0:22
10	**200**	8.29	10.33	2.12	0.63	2.21	3.92	3.29	2.19	2.08	1.67	1.42	1.25	1.50	1.05	1.80	1.17	2	2	1	2	1	1	2	1	0:35	1:23	0:09	0:11	0:05	0:04	0:13	0:33
20	**0**	8.29	11.53	2.85	1.07	1.65	6.57	4.32	2.62	1.46	1.23	1.21	1.21	1.77	1.00	1.60	1.11	2	2	1	1	3	0	3	1	0:40	1:12	0:07	0:10	0:04	0:04	0:11	0:20
20	**10**	8.70	12.22	3.36	1.17	1.77	5.11	3.57	2.45	1.56	1.15	1.29	1.21	1.73	1.05	1.63	1.14	2	2	2	1	2	1	2	1	0:39	1:07	0:07	0:11	0:04	0:03	0:12	0:31
20	**50**	9.28	11.61	4.41	2.17	2.25	6.16	3.50	2.65	1.67	1.36	1.21	1.25	1.32	1.05	1.51	1.14	2	2	1	1	1	1	2	1	0:39	1:04	0:07	0:11	0:05	0:05	0:12	0:20
20	**100**	9.44	13.44	2.59	1.02	1.86	7.35	4.87	2.38	1.59	1.13	1.25	1.25	1.55	1.05	1.37	1.20	2	1	1	1	1	1	1	1	0:38	1:09	0:07	0:12	0:05	0:05	0:12	0:40
20	**200**	11.80	9.80	3.58	1.00	1.76	4.92	5.27	2.52	1.33	1.15	1.21	1.25	1.45	1.00	1.43	1.11	1	1	1	1	1	0	1	1	0:37	1:06	0:07	0:10	0:05	0:03	0:12	0:28
50	**0**	12.20	15.11	5.12	1.15	3.77	8.21	5.97	2.95	1.23	1.00	1.21	1.21	1.55	1.00	1.37	1.03	2	0	2	1	4	0	2	1	0:36	1:02	0:07	0:08	0:05	0:03	0:11	0:37
50	**10**	12.03	17.58	3.14	2.42	4.17	8.34	6.68	3.34	1.33	1.05	1.33	1.21	1.55	1.09	1.34	1.06	2	1	3	1	2	1	3	1	0:38	0:48	0:08	0:08	0:06	0:02	0:12	0:32
50	**50**	10.69	12.86	4.71	1.07	2.53	6.42	6.71	3.83	1.15	1.00	1.58	1.21	1.36	1.05	1.29	1.03	2	0	2	1	2	1	1	1	0:38	0:45	0:08	0:08	0:06	0:03	0:12	0:23
50	**100**	10.85	15.34	2.51	1.53	3.47	6.01	6.98	3.14	1.08	1.00	1.42	1.29	1.14	1.00	1.29	1.06	1	0	2	1	1	0	1	1	0:37	0:49	0:08	0:08	0:06	0:02	0:12	0:31
50	**200**	11.97	14.79	4.56	1.08	2.33	5.10	6.71	3.60	1.13	1.03	1.21	1.21	1.23	1.05	1.29	1.00	1	1	1	1	1	1	1	0	0:39	0:49	0:08	0:07	0:07	0:02	0:12	0:36
100	**0**	13.03	20.61	3.10	2.86	4.50	9.28	8.82	3.49	1.05	1.00	1.33	1.21	1.14	1.00	1.17	1.00	1	0	1	1	2	0	1	0	0:40	0:42	0:07	0:05	0:06	0:02	0:13	0:18
100	**10**	12.21	21.52	1.75	1.42	4.23	9.68	9.43	3.65	1.31	1.03	1.46	1.21	1.27	1.00	1.20	1.00	2	1	2	1	2	0	1	0	0:37	0:47	0:07	0:06	0:06	0:02	0:12	0:13
100	**50**	10.55	23.29	1.75	1.42	4.82	7.04	9.06	4.06	1.18	1.00	1.46	1.21	1.27	1.05	1.17	1.00	1	0	2	1	1	1	1	0	0:38	0:44	0:07	0:06	0:05	0:02	0:12	0:23
100	**100**	11.69	15.11	2.63	1.37	4.24	11.80	9.43	4.33	1.15	1.00	1.25	1.21	1.27	1.00	1.20	1.00	1	0	1	1	1	0	1	0	0:42	0:46	0:07	0:06	0:06	0:02	0:12	0:33
100	**200**	10.96	17.26	2.63	1.42	4.75	12.31	9.43	3.98	1.21	1.00	1.25	1.21	1.23	1.00	1.20	1.00	1	0	1	1	1	0	1	0	0:36	0:40	0:07	0:05	0:05	0:02	0:12	0:16
200	**0**	16.44	18.90	2.54	5.37	3.22	9.89	13.29	5.10	1.10	1.00	1.42	1.17	1.23	1.05	1.00	1.03	1	0	3	1	2	1	0	1	0:36	0:45	0:08	0:05	0:05	0:02	0:11	0:14
200	**10**	19.62	14.42	3.86	1.12	12.62	15.31	11.67	4.62	1.03	1.00	1.21	1.21	1.05	1.05	1.03	1.00	1	0	1	1	1	1	1	0	0:38	0:45	0:08	0:05	0:05	0:02	0:11	0:14
200	**50**	14.35	21.13	3.41	1.34	11.69	8.79	13.48	4.29	1.05	1.00	1.29	1.25	1.05	1.05	1.00	1.00	1	0	1	1	1	1	0	0	0:40	0:40	0:08	0:04	0:06	0:02	0:11	0:21
200	**100**	13.56	22.61	3.02	4.86	12.14	8.79	12.58	4.28	1.08	1.00	1.21	1.25	1.00	1.05	1.03	1.00	1	0	1	1	0	1	1	0	0:37	0:42	0:08	0:05	0:06	0:02	0:11	0:20
200	**200**	14.07	19.49	2.53	3.42	9.66	17.25	10.16	4.09	1.08	1.00	1.21	1.21	1.00	1.05	1.00	1.00	1	0	1	1	0	1	0	0	0:38	0:45	0:07	0:05	0:06	0:02	0:11	0:16

Figure 3 and Figure 4 show the results for respectively the average spreading costs and the average number of exam versions. Each row of graphs represents a different dataset and each column represents a different value of c^N. Our goal was to develop an examination-timetabling approach capable of trading off multiple exam versions with lower spreading costs. Figure 3 shows that in most cases, the average student spreading cost increases for higher exam version cost in both models. The opposite reaction for the average spreading costs is exhibited in Figure 4.

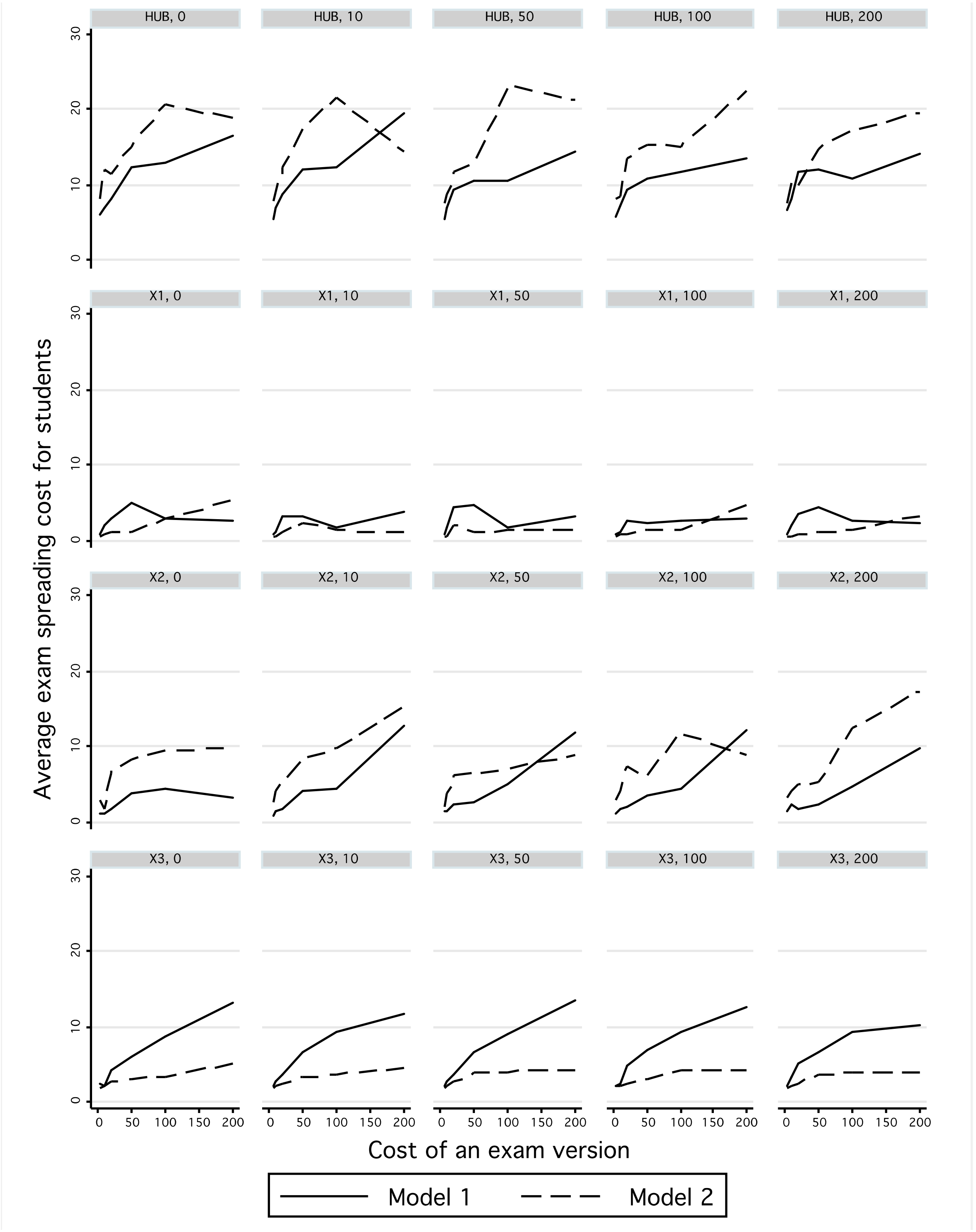


Figure 3 Results for the average student spreading cost in function of the cost of one exam version; each row shows the graphs for the same dataset but for a different value of c^N (with c^N represented in the columns)

Generally, as the number of exam versions decreases with rising exam version costs, the spreading costs for students increase, which is the expected trade-off. Both models, however, have their strong point: while the first model somewhat favors lower spreading costs, the second model visibly favors fewer exam versions. Table 2 shows similar results for the maximum number of additional exam versions N. Model 2 generally produces timetables with lower or equal values of N. We believe this is due to the post-processing phase in the second model, which adds extra flexibility.

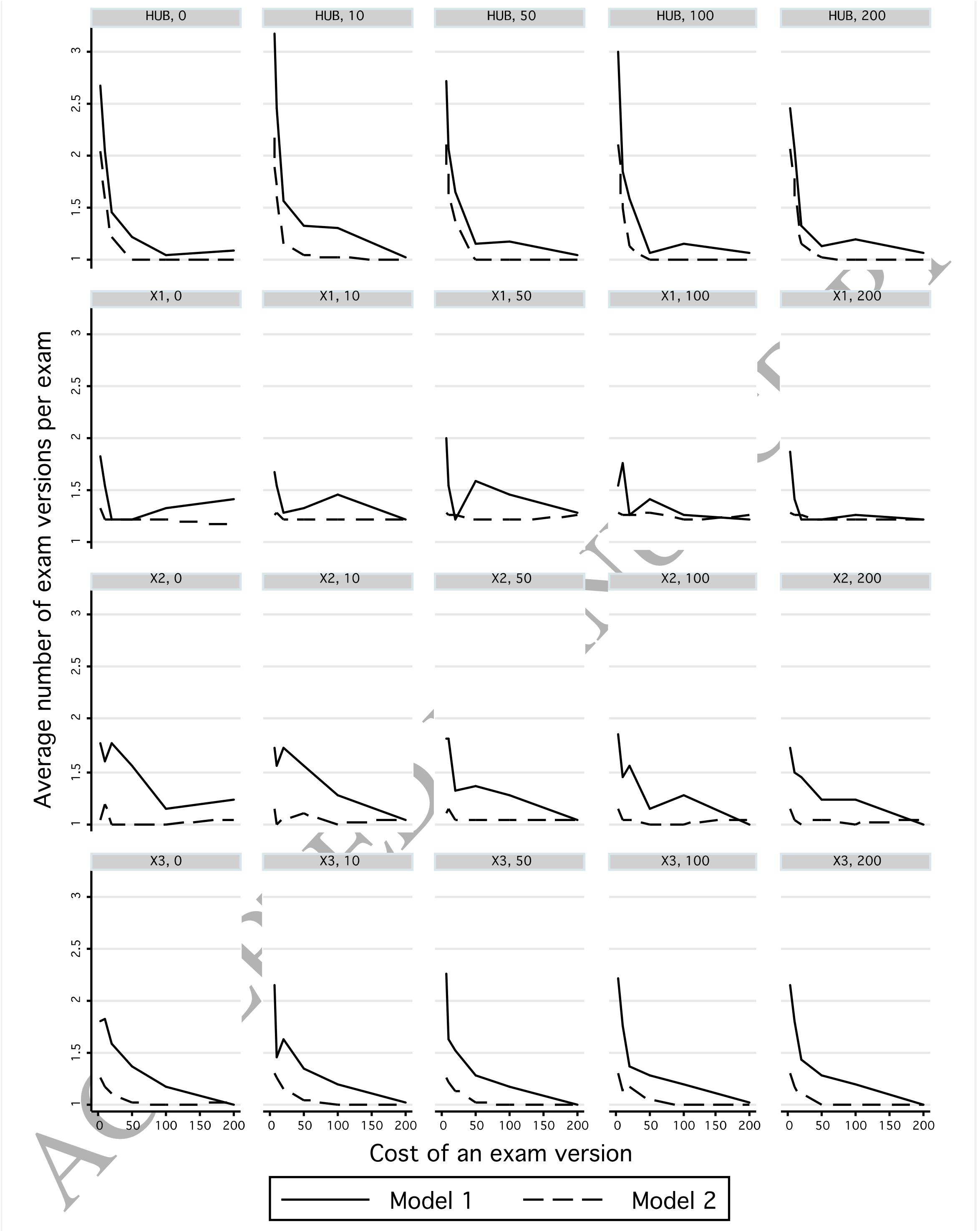


Figure 4 Results for average number of exam versions in function of the cost of one exam version; each row shows the graphs for the same dataset but for a different value of c^N (with c^N represented in the columns)

To assess the overall performance of the algorithm, we compare the total cost. Figure 5 shows the integer goal function values for the instances *HUB*, *X1*, *X2*, and *X3*. In general, Model 2 produces timetables with lower total costs and has a performance advantage of 5.2% when compared to the first model. For datasets *X1* and *X3*, Model 2 results in a lower cost of 15% and 26% respectively when compared to the first model. However, for the *HUB* and *X2* datasets, Model 2 has a performance disadvantage of 6% and 14% respectively when compared to the first model.

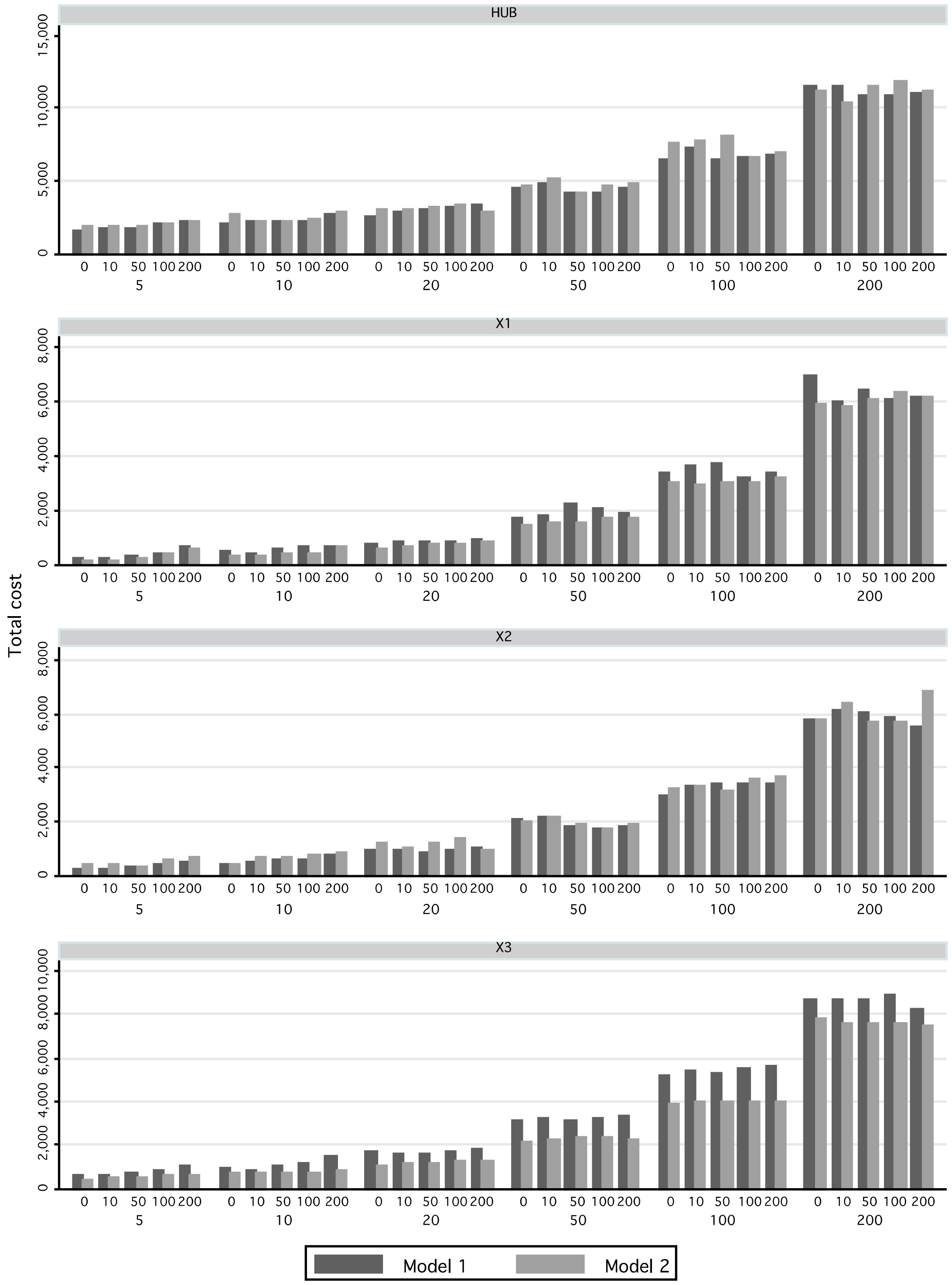


Figure 5 Integer goal function values for datasets *HUB*, *X1*, *X2*, and *X3* for various parameters of c^N (upper values on the X-axis) and c^V (lower values on the X-axis) as categories

From these results, we conclude that the second model slightly outperforms the first one in terms of solution quality. Both models however have different CG processes. Figure 6 and Figure 7 show the typical CG process for respectively Model 1 and Model 2. The graphs represent the relaxed goal function value of the RMP and the Lagrangian lower bound. The graph for the second model clearly displays the tailing-off effect that is typically associated for CG each time the CG process is restarted after fixing a series of examination dates. We remedied this by aggregation in Model 1 and by using a stop criterion for Model 2. Both graphs show spikes in the goal function value when fixing examination time slots. As a result Model 1 is notably faster than Model 2, due to the absence of the tailing-off effect.

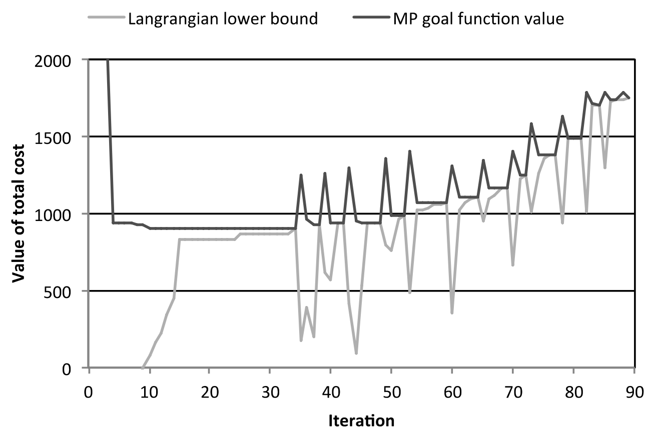


Figure 6 Example of a CG process for Model 1 for the HUB dataset using $c^V = 10$ and $c^N = 100$

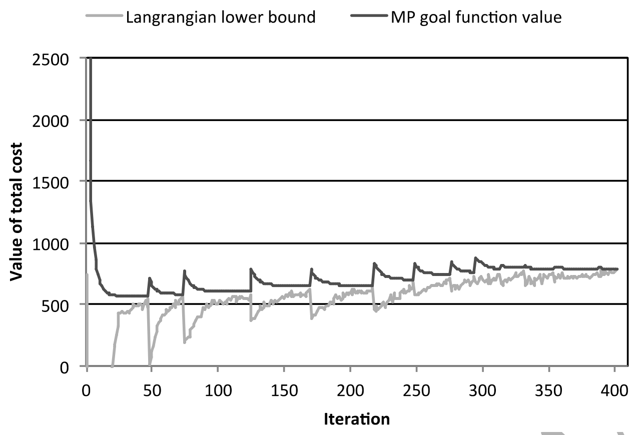


Figure 7 Example of a CG process for Model 2 for the HUB dataset using $c^V = 10$ and $c^N = 100$

We also test the second model using the post-processing heuristic. The results of the heuristic are compared to IP as post-processing technique in Table 3, which presents the resulting number of versions, the maximum number of extra exam versions, and the total cost for exams for dataset *X1* with parameters $c^V = 10$ and $c^N = 100$. The results for other instances are similar. In general, the heuristic results in only a slightly different result as compared to integer programming as post-processing technique. The heuristic has a higher number of maximum additional exam versions and the same number of average exam versions.

Table 3
Comparison of the post-processing techniques for dataset *X1* with parameters $c^V = 10$ and $c^N = 100$

Post-processing technique	Maximum number of extra versions (N)	Average number of versions	Total cost for exams
Heuristic	2	1.25	500
IP	1	1.25	400

5.3 Test results for the literature datasets

We test our models on two existing instances from the *Toronto* dataset (Carter et al., 1996) : *STA83* and *YOR83* with different parameters values. For the second model, we used the IP post-processing technique. Model 1 takes between 1.5 and 2.5 hours to solve an instance of the *STA83* set and between 13 and 15 hours for an instance of the *YOR83* set, while the second model needs between 2 and 8 hours to solve an instance of the *STA83* set. Model 2 was not able to solve the larger *YOR83* set due to computational time constraints. **Table 4** summarizes the results.

Model 1 produces results within the specified time limit for all parameters and datasets, while Model 2 is only able to do this for the *STA83* dataset, but not for the instances of the *YOR83* dataset. Model 2's MP is very time-consuming for datasets of this size and is unable to reach the step that fixes examination dates.

The first model is sensitive to the cost parameters relative to the size of the problem, to produce a good trade-off. Cost parameters need to be sufficiently large, for the trade-off to be visible. In the existing literature, the best average spreading costs, while scheduling each exam only once, are 156.9 (Burke et al., 2010) for the *STA83* set and 34.64 (Demeester et al., 2012) for the *YOR83* set. For the first dataset, the models show a small improvement of up to 3% in spreading cost by allowing more exam versions to be scheduled. For the second dataset, this was 7%. The second model produces timetables with a lower total cost as compared to the first model for the *STA83* set.

Table 4
Spreading costs, relative improvement relative to the best reported student spreading cost, average number of versions per exam and maximum number of extra versions for the *Toronto* instances *STA83* and *YOR83*, * means the instance did not solve within 400 hours

Metric:			**Average student spreading cost**		**Difference in average spreading cost relative to best reported**		**Average number of versions**		**Maximum number of extra versions (N)**		**Integer goal function value**	
Model:			**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**
Dataset	c^V	c^N										
STA83	**10**	**100**	152.55	152.25	-3%	-3%	2.24	2.07	5	4	96,819	96,302
STA83	**50**	**100**	154.05	154.56	-2%	-1%	1.60	1.09	4	1	105,672	102,112
STA83	**100**	**100**	155.73	156.73	-1%	0%	1.78	1.03	6	1	230,448	110,179
STA83	**200**	**200**	158.92	157.78	1%	+1%	1.44	1.012	4	1	137,902	124,682
STA83	**1000**	**5000**	153.00	157.78	-2%	+1%	2.15	1.007	5	1	417,480	241,402
YOR83	**100**	**100**	33.28	*	-4%	*	3.04	*	12	*	87,516	*
YOR83	**100**	**1000**	32.14	*	-7%	*	3.24	*	9	*	98,045	*
YOR83	**1000**	**1000**	36.33	*	5%	*	3.06	*	11	*	599,186	*
YOR83	**1000**	**100000**	35.77	*	3%	*	3.09	*	8	*	1,393,664	*

5.4 Discussion

Both models are successful in making a trade-off between spreading and exam costs and thus in creating high-quality examination timetables, especially for smaller-sized test instances. The results show a good response of the models to changing parameters. For the existing *STA83* set, both models are able to give a better spreading cost to students as compared to the best solution found in the literature, but not for all instances and at the cost of multiple exam versions. The second model performed better in terms of total costs. For the larger *YOR83* set, only the first model was able to obtain a solution, whereas the second model exceeded the CPU-time limit of 400 hours. These results show that our models are well suited for smaller-sized problems such as the *HUB* datasets, but are unsuited to solve larger problems such as the *YOR83* dataset. The number of student groups increases the number of columns that are generated for each iteration, increasing the size of the MP significantly. Our heuristic method of fixing examination dates also appeared to be unsuited to address the fractional solutions from the CG-process with problems of this size. We believe this is due to the increase in potential conflicts, and the chain of potential conflicts with other exams. The increase in fractional solutions and the nature of the current fixing process makes it more difficult to find solutions with no additional exam versions for large datasets. Note that for small datasets it is possible to find solutions that schedule exams only once.

The difference between the two models underlines the decision of which complexity to transfer to the PP. Without a branch-and-price algorithm that guarantees an optimal solution, the second model's MP has increased flexibility in finding solutions because of a reduced transfer in complexity. Whereas the first model is more straightforward and is always able to produce feasible timetables, the second model produces timetables with lower total costs. The relaxed MP produces fractional solutions, resulting in an artificially low number of versions for every exam. In the integer MP, this was of course not allowed. As our fixing of variables in the MP is performed heuristically, Model 2 benefits from the increased flexibility it offers during the post-processing for planning the final exam versions. Model 1 is constrained by the scheduled exams in the generated columns that cannot be changed afterwards. Our attempt to counteract the tailing-off effect typically associated with CG (by using an aggregated constraint formulation in Model 1) was successful. The use of a Lagrangian bound stopping-criterion in the second model, does limit time consumption, but does not resolve the tailing-off effect.

6 Conclusion

In this article, we look at examination timetabling from a student-centric point of view. Instead of scheduling an exam once while optimizing the exam spread for students, we optimize the exam spread by scheduling exams multiple times for different student groups if this is sufficiently beneficial for the exam spread. The cost of the exam spread is measured using the method developed by Laporte and Desroches (1984). We optimize a trade-off function by weighing the costs of better spreading for a student versus the cost of scheduling multiple versions of an exam. To our knowledge, no other author has designed models to make a trade-off between exam spreading and the number of exam versions, even though this is quite common in colleges and universities. We propose two Column Generation (CG) models to solve the Examination-Timetabling Problem (ETP). The objective is to schedule exams in time slots, so that the required number of exam versions is minimal while maintaining a fair schedule for each student. Each model uses a different definition of a column that serves as a schedule for a unique student group.

The first model defines a column to be a complete examination timetable for a unique student group. CG requires the integrality constraint to be relaxed during the generation of columns. The first model counters fractional solutions by heuristically fixing exams at certain time slots during the CG process, based on the number of potential conflicts with other exams. It allows finding new columns for other student groups while taking into account the already fixed columns and their scheduled exams.

The second model uses mask slot schedules as columns. These mask slot schedules define when students will take an exam. The Master Program (MP) decides which exam is scheduled at what time slot, and implicitly schedules exam versions for student groups. Both models fix exams at certain time slots and allow for finding new columns that take the already fixed exams into account. Model 2, however, needs post-processing to schedule all exams for all students. We developed an IP as well as a heuristic for this post-processing phase.

We test our models on both real-life data from KU Leuven campus Brussels (Belgium) and a number of randomly generated test instances. Both models find high-quality solutions for all instances of our own datasets within an acceptable time limit, with the second model producing the timetables with the lowest total cost. Model 1 has a tendency to favor low spreading costs over fewer exam versions, resulting in higher total costs. Some performance concerns remain with respect to solving large-scale ETPs. The algorithms are applied to two datasets from the *Toronto b* problem: the *STA83* set and the *YOR83* set. The first model is able to find good solutions for both datasets for various cost parameters, while the second model was only able to find a solution for the *STA83* set within the given time limit. The models are able to produce timetables that improve spreading costs for students, at the cost of additional exam versions. The size of the problem hinders the MP and also hinders the current method of fixing examination dates to adhere to integrality constraints. We conclude that for larger dimensions our models are not suited, especially when solutions with no additional exams are required. The results show that spreading costs can be strongly reduced by allowing multiple versions of exams.

Both CG models would benefit strongly from an efficient and exact branch-and-price algorithm that would extend the current models. Other possible research directions include to speed up the proposed algorithm by focusing on better starting solutions and alternative pricing rules to aid the CG process. Our models can easily be extended to include new requirements to the pricing problems. Requirements that apply within a group's schedule and that are unrelated to specific exams can easily be added to the pricing problem of both Model 1 and Model 2. For example, one could specify that a specific student group can have no more than two exams in one week due to a learning disability. It might also be fruitful to study the adaptability of the existing solution methods from the literature to incorporate the trade-off presented in this article.

To summarize, in this article we implemented the common practice of scheduling exams multiple times, which is beneficial to the spreading costs for students. We developed two CG algorithms and compared them using a computational experiment in which the second algorithm outperformed the first one. The results described in this article indicate that CG could be a valuable method in solving the ETP by means of decomposition.

References

Abdul-Rahman, S., Burke, E.K., Bargiela, A., McCollum, B., Özcan, E., 2014a. A constructive approach to examination timetabling based on adaptive decomposition and ordering. Ann. Oper. Res. 218(1), 3-21.

Abdul-Rahman, S., Bargiela, A., Burke, E.K., McCollum, B., Özcan, E., McMullan, P., 2014b. Adaptive linear combination of heuristic orderings in constructing examination timetables. Eur. J. Oper. Res. 232(2), 287-297.

Al-Betar, M.A., Khader, A.T., Abu Doush, I., 2014. Memetic techniques for examination timetabling. Ann. Oper. Res. 218(1), 23-50.

Al-Yakoob, S., Sherali, H.D., Al-Jazzaf, M., 2010. A mixed-integer mathematical modeling approach to exam timetabling. Comput. Manag. Sci. 7, 19–46. doi:10.1007/s10287-007-0066-8

Alzaqebah, M., Abdullah, S. 2014. An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling. J. Sched. 17(3), 249-262.

Alzaqebah, M., Abdullah, S. 2015. Hybrid bee colony optimization for examination timetabling problems. Comput. Oper. Res. 54, 142-154.

Arbaoui, T., Boufflet, J.P., Moukrim, A., 2015. Preprocessing and an improved MIP model for examination timetabling. Ann. Oper. Res. 229(1), 19-40.

Azimi, Z.N., 2004. Comparison of metaheuristic algorithms for examination timetabling problem. J. Appl. Math. Comput. 16, 337–354. doi:10.1007/BF02936173

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H., 1998. Branch-and-price: column generation for solving huge integer programs. Oper. Res. 46, 316–329. doi:10.1287/opre.46.3.316

Bosch, R., Trick, M., 2005. Integer Programming, in: Burke, E., Kendall, G. (Eds.), Search Methodologies SE - 3. Springer US, pp. 69–95. doi:10.1007/0-387-28356-0_3

Bullnheimer, B., 1998. An examination scheduling model to maximize students' study time, in: Burke, E., Carter, M. (Eds.), Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science. Springer, Berlin, pp. 78–91. doi:10.1007/BFb0055882

Burke, E.K., Dror, M., Petrovic, S., Qu, R., 2005. Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems, in: Golden, B., Raghavan, S., Wasil, E. (Eds.), The next Wave in Computing, Optimization, and Decision Technologies, Operations Research/Computer Science Interfaces Series. Springer US, New York, NY, pp. 79–91. doi:10.1007/0-387-23529-9_6

Burke, E.K., Eckersley, A.J., McCollum, B., Petrovic, S., Qu, R., 2010. Hybrid variable neighbourhood approaches to university exam timetabling. Eur. J. Oper. Res. 206, 46–53. doi:10.1016/j.ejor.2010.01.044

Burke, E.K., Jackson, K., Kingston, J., Weare, R., 1997. Automated University Timetabling: The State of the Art. Comput. J. 40, 565–571.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R., 2007. A graph-based hyper-heuristic for educational timetabling problems. Eur. J. Oper. Res. 176, 177–192. doi:10.1016/j.ejor.2005.08.012

Burke, E.K., Newall, J.P., Weare, R.F., 1998. Initialization Strategies and Diversity in Evolutionary Timetabling. Evol. Comput. 6, 81–103. doi:10.1162/evco.1998.6.1.81

Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. Eur. J. Oper. Res. 140, 266–280. doi:10.1016/S0377-2217(02)00069-3

Burke, E.K., Qu, R., Soghier, A., 2014. Adaptive selection of heuristics for improving exam timetables. Ann. Oper. Res. 218(1), 129-145.

Carter, M.W., 1986. A survey of practical applications of examination timetabling algorithms. Oper. Res. 34, 193–202. doi:10.1287/opre.34.2.193

Carter, M.W., Laporte, G., Lee, S.Y., 1996. Examination Timetabling: Algorithmic Strategies and Applications. J. Oper. Res. Soc. 47, 373. doi:10.2307/3010580

Cole, A.J., 1964. The preparation of examination time-tables using a small-store computer. Comput. J. 7, 117–121. doi:10.1093/comjnl/7.2.117

Cooper, T., Kingston, J., 1996. The complexity of timetable construction problems, in: Burke, E., Ross, P. (Eds.), Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 281–295. doi:10.1007/3-540-61794-9_66

Demeester, P., Bilgin, B., De Causmaecker, P., Van Den Berghe, G., 2012. A hyperheuristic approach to examination timetabling problems: Benchmarks and a new problem from practice. J. Sched. 15, 83–103. doi:10.1007/s10951-011-0258-5

Dimopoulou, M., Miliotis, P., 2001. Implementation of a university course and examination timetabling system. Eur. J. Oper. Res. 130, 202–213. doi:10.1016/S0377-2217(00)00052-7

Dowsland, K.A., Thompson, J.M., 2004. Ant colony optimization for the examination scheduling problem. J. Oper. Res. Soc. 56, 426–438. doi:10.1057/palgrave.jors.2601830

Gogos, C., Alefragis, P., Housos, E., 2010. An improved multi-staged algorithmic process for the solution of the examination timetabling problem. Ann. Oper. Res. 194, 203–221. doi:10.1007/s10479-010-0712-3

Kahar, M.N.M., Kendall, G., 2010. The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. Eur. J. Oper. Res. 207, 557–565. doi:10.1016/j.ejor.2010.04.011

Kahar, M.N.M., Kendall, G., 2014. A great deluge algorithm for a real-world examination timetabling problem. J. Oper. Res. Soc. 66(1), 116-133.

Kendall, G., Hussin, N., 2005. An investigation of a tabu-search-based hyper-heuristic for examination timetabling, in: Kendall, G., Burke, E.K., Petrovic, S., Gendreau, M. (Eds.), Multidisciplinary Scheduling: Theory and Applications. Springer US, New York, NY, pp. 309–328. doi:10.1007/0-387-27744-7_15

Komar, M., Grbic, D., Cupic, M., 2011. Solving exam timetabling using distributed evolutionary computation, in: Luzar-Stiffler, V., Jarec, I., Bekic, Z. (Eds.), 2011 33rd International Conference on Information Technology Interfaces. IEEE, Piscataway, NJ, USA, pp. 301–306.

Laporte, G., Desroches, S., 1984. Examination timetabling by computer. Comput. Oper. Res. 11, 351–360. doi:10.1016/0305-0548(84)90036-4

Li, J., Bai, R., Shen, Y., Qu, R., 2015. Search with evolutionary ruin and stochastic rebuild: A theoretic framework and a case study onexam timetabling. Eur. J. Oper. Res. 242(3), 798-806.

Liu, Y., Zhang, D., Chin, F.Y.L., 2011. A clique-based algorithm for constructing feasible timetables. Optim. Methods Softw. 26, 281–294. doi:10.1080/10556781003664739

Lübbecke, M.E., Desrosiers, J., 2005. Selected topics in column generation. Oper. Res. 53, 1007–1023. doi:10.1287/opre.1050.0234

Malik, A.M.A., Othman, A.K., Ayob, M., Hamdan, A.R., 2011. Hybrid integrated two-stage multi-neighbourhood tabu search-EMCQ technique for examination timetabling problem, in: Hamdan, A.R., Famili, F.A., Kendall, G., Sarim, H.M., Osman, I.H., Abdullah, S., Othman, Z. (Eds.), 2011 3rd Conference on Data Mining and Optimization (DMO 2011). IEEE, Piscataway, NJ, USA, pp. 232–236.

Mansour, N., Sleiman-Haidar, G., 2011. Parallel Scatter Search Algorithms for Exam Timetabling. Int. J. Appl. Metaheuristic Comput. 2, 27–44.

Marie-Sainte, S.L., 2015. A survey of Particle Swarm Optimization techniques for solving university Examination Timetabling Problem. Artif. Intell. Rev. 44(4), 537-546.

Mehrotra, A., Trick, M.A., 1996. A column generation approach for graph coloring. INFORMS J. Comput. 8, 344–354. doi:10.1287/ijoc.8.4.344

Mehta, N.K., 1981. The application of a graph coloring method to an examination scheduling problem. Interfaces (Providence). 11, 57–65. doi:10.1287/inte.11.5.57

Pais, T.C., Amaral, P., 2011. Managing the tabu list length using a fuzzy inference system: an application to examination timetabling. Ann. Oper. Res. 194, 341–363. doi:10.1007/s10479-011-0867-6

Qu, R., Burke, E.K., McCollum, B., 2009. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. Eur. J. Oper. Res. 198, 392–404. doi:10.1016/j.ejor.2008.10.001

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G., Lee, S.Y., 2008. A survey of search methodologies and automated system development for examination timetabling. J. Sched. 12, 55–89. doi:10.1007/s10951-008-0077-5

Qu, R., Nam P., Bai, R., Kendall, G., 2015. Hybridising heuristics within an estimation distribution algorithm for examination timetabling, Appl. Intell. 42(4), 679-693.

Qualizza, A., Serafini, P., 2005. A column generation scheme for faculty timetabling, in: Burke, E., Trick, M. (Eds.), Practice and Theory of Automated Timetabling V, Lecture Notes in Computer Science. Springer, Berlin, pp. 161–173. doi:10.1007/11593577_10

Queen's University of Belfast, 2007. International Timetabling Competition 2007 [WWW Document]. URL http://www.cs.qub.ac.uk/itc2007/index.htm (accessed 5.31.14).

Sabar, N.R., Ayob, M., Kendall, G., Qu, R., 2012. A honey-bee mating optimization algorithm for educational timetabling problems. Eur. J. Oper. Res. 216, 533–543. doi:10.1016/j.ejor.2011.08.006

Sabar, N.R., Ayob, M., Qu, R., Kendall, G., 2011. A graph coloring constructive hyper-heuristic for examination timetabling problems. Appl. Intell. 37, 1–11. doi:10.1007/s10489-011-0309-9

Schaerf, A., 1999. A survey of automated timetabling. Artif. Intell. Rev. 13, 87–127. doi:10.1023/A:1006576209967

Soghier, A., Qu, R., 2013. Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. Appl. Intell. 39, 438–450. doi:10.1007/s10489-013-0422-z

Vanderbeck, F., 2005. Implementing Mixed Integer Column Generation, in: Desaulniers, G., Desrosiers, J., Solomon, M. (Eds.), Column Generation SE - 12. Springer US, pp. 331–358. doi:10.1007/0-387-25486-2_12

White, G.M., Xie, B.S., Zonjic, S., 2004. Using tabu search with longer-term memory and relaxation to create examination timetables. Eur. J. Oper. Res. 153, 80–91. doi:10.1016/S0377-2217(03)00100-0

Wijgers, R., Hoogeveen, H., 2007. A column generation approach for examination timetabling (No. UU-CS-2007-001), Technical report. Utrecht: Universiteit Utrecht.

Zhang, D., Liu, Y., M'Hallah, R., Leung, S.C.H., 2010. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. Eur. J. Oper. Res. 203, 550–558. doi:10.1016/j.ejor.2009.09.014