The student scheduling problem at Université de Technologie de Compiègne

Jean-Paul Boufflet, Taha Arbaoui, Aziz Moukrim

The Student Scheduling Problem at Université de Technologie de Compiègne

Jean-Paul Boufflet[a,*], Taha Arbaoui[b], Aziz Moukrim[a]

[a] *Sorbonne universités, Université de Technologie de Compiègne*
Laboratoire Heudiasyc, UMR CNRS 7253
57 avenue de Landshut CS 60319
60203 Compiègne, France
[b] *Université de Technologie de Troyes*
Laboratoire d'Optimisation des Systèmes Industriels
Institut Charles Delaunay (ICD-LOSI), FRE CNRS 2019
12 rue Marie Curie, CS 42060
10004, Troyes, France

[*] Corresponding author. Tel: +333 44 23 46 91; Fax: +333 44 23 44 77
Email addresses: jean-paul.boufflet@hds.utc.fr (Jean-Paul Boufflet), taha.arbaoui@utt.fr (Taha Arbaoui), aziz.moukrim@hds.utc.fr (Aziz Moukrim)

Preprint submitted to Elsevier

Abstract

At Université de Technologie de Compiègne (UTC), complete course timetables are made available to students for the forthcoming semester, then students select their courses. A student cannot attend two events at the same time without violating a hard conflict constraint. Ideally, a conflict-free individual timetable will be created for each student, but it may be the case that there is no feasible solution for all students simultaneously. With thousands of students and hundreds of courses, UTC's Student Scheduling Problem (SSP) cannot be processed manually. A heuristic designed decades ago can no longer be used, because too many students are left non-assigned and because the heuristic was not designed to deal with the current requirements. In this study we propose a preprocessing that reduces the size of instances, together with lower bounds that allow the quality of computed solutions to be assessed. Where most of the graphs relating to students' hard conflict constraints are interval graphs, we show that a clique formulation of the hard conflict constraints may substantially reduce the number of equations in relation to other formulations. We propose integer linear programming (ILP) formulations to address the current requirements. We test an ILP with a global objective function and other ILPs within a lexicographic scheme. We investigate valid inequalities for reducing computation times. We report our computational experiments and results obtained on real instances from UTC. The solution method has proved its effectiveness and is now the tool used for student scheduling at UTC.

keyword: University timetabling, Preprocessing, Integer Programming, Valid inequalities, Lexicographical optimization

1. Introduction

For decades, course and examination timetabling problems in schools and academic institutions have been the subject of intensive research. This is because of the wide variety of problems of practical interest, given institutions' differing organizational requirements, and because usually these are hard optimization problems that can be addressed using a wide range of methods.

In our university (UTC), course timetables are first drawn up, and students can consult these when choosing the courses for which they wish to enroll. Once students' choices are known, in order to provide each student with an individual timetable a Student Scheduling Problem (SSP) with multiple objective criteria relating to our institution's requirements needs to be addressed.

At UTC, a course is known as a *Unité de Valeur* (UV) and corresponds to a total volume of working hours over a semester. The work to be undertaken by the student will involve a variety of learning activities (*activity*) including attending lectures, taking part in tutorials and meetings, and doing labs (i.e., practical sessions). For the purposes of these learning activities the students enrolled on a UV will be assigned to a *section* for each activity. Lectures usually have a single section, while for other activities students will usually be assigned

among different sections (smaller groups scheduled to do the activity at different times). Each section has at least one *event* scheduled either weekly or fortnightly and with a set duration (between one and four hours). Most events take place in a room with a limited capacity. As an example, there are UVs that have one weekly lecture section (2-hour event, 144 students), six weekly tutorial sections (2-hour event, 24 students) and twelve fortnightly lab sections (4-hour event, 12 students). The sections' sizes are determined according to the type of activity and the available resources (number of seats, computers, lab benches, etc.). The timetables made available to students at the time of enrollment include the rooms and the time slots for the events scheduled for the different sections. Thus, students are able to take this information into account in their choice of UVs. Today, there are on average 2500 students to be assigned to 1400 sections. Instances are therefore too large to be processed manually, and the number of students is also expected to rise in the coming years.

A student must be assigned to one section for each of the learning activities that constitute his/her chosen courses. Our university does not allow a student to be assigned to more than one event at the same time. This is a strict institutional constraint (hard conflict constraint). Moreover, as long as a student has not been assigned to a section for all the learning activities in which he/she is required to participate, that student is deemed *non-assigned* and is not allowed to undertake the UV. This is also a strict constraint.

Even when a there is a possible assignment to sections that *for each student taken individually* is compatible with the different course timetables, there might not be a feasible solution for all students simultaneously (for example, because of the capacities of rooms). A heuristic created decades ago was, for many years, used to assign students to sections, but tens of students were left non-assigned every semester, and it was simply not known whether an alternative more complete assignment might exist. This heuristic has now been abandoned because of its deficiency in this respect and because of new requirements by the university that the heuristic failed to address.

Until recently, all students had to be assigned to a section for *every* activity included in their chosen set of UVs. Now, in a drive to reduce operating costs, the university will sometimes exempt students who retake a UV from having to do all of the activities a second time. For example, when a student retakes a UV that includes labs, provided that grades for the labs were deemed acceptable the first time around, the student may be given an exemption for labs and not be required to be assigned to a lab section a second time (which saves consumables).

Moreover, the university is attempting to provide a more flexible curriculum in order to respond better, for example, to the needs of students with disabilities and the needs of industrial sandwich students.

Given the changing context, our university has defined new requirements. The first requirement is a more successful minimizing of the number of *non-assigned* students, on the basis of *sets of activities* rather than *sets of UVs*. The second and third requirements are minimizing the number of *hurried* moves (i.e., moves for which little time is available) *between sites* and *between buildings on the same site* for students with reduced mobility. The fourth requirement is

minimizing the number of sections into which particular populations of students are assigned. The objective, in some activities, of assigning certain subsets of students to as few separate sections as possible has arisen from a changing curriculum and from the needs of students with disabilities. The fifth requirement is minimizing the number of hurried moves between sites for students in general. UTC has placed these new requirements in the order of priority given above, and this order is reflected in our SSP criteria.

In this study, we report our works and experiments related to the Student Scheduling Problem (SSP) that we face every semester. Given that tens of students are non-assigned using a heuristic approach, we have chosen to investigate an exact approach. However, the more criteria there are, the larger the instance to be solved becomes and computation times can be in some cases too large. We also consider ways of reducing the size of the instances, different formulations designed to reduce the number of constraints, and valid inequalities to reduce computation times.

The contributions of this work can be summarized as follows:

- We tackle the practical SSP at UTC that requires a specific combination of hard and soft constraints.
- We propose a preprocessing that proved to be effective in reducing the size of instances.
- We derive lower bounds for the minimum number of moves and the minimum number of sections to be used for grouping students together, against which the solutions that we obtain can be compared.
- We introduce a new clique-based formulation for the hard conflict constraints. This formulation substantially reduces the number of equations in relation to other formulations where most of the graphs relating to students' hard conflict constraints are interval graphs.
- We propose an ILP formulation seeking to minimize a global objective function, and ILP formulations within a lexicographic optimization scheme to address a series of criteria in turn.
- We investigate valid inequalities with the aim to speed up computation times and to help to attain optimal solutions in cases where there are larger numbers of students that cannot be assigned.
- Using fifteen real-world problem instances, we show how the lexicographic optimization scheme can be used to overcome the difficulty of solving the UTC's SSP considering all the terms of the objective function while attaining optimal solutions.

The entire process can be adapted to solve other real-world standalone SSP.

Structure of the paper: Section 2 is a review of the literature relating to student scheduling problems. Section 3 outlines the supported process and

the heuristic used thus far, and details the current institutional requirements together with data and parameters. Section 4 presents our proposed preprocessing for reducing the size of instances, and the lower bounds that we established for criteria related to moving and grouping. We discuss the conflict graph for a student and the condition that gives us a reasonable number of clique-based equations for enforcing conflict constraints. Section 5 details the ILP model with a global objective function, our lexicographic optimization, and the valid inequalities that we used in our experiments. Section 6 reports our computational experiments with our university's real-world data and discusses the results we obtained. Conclusions and future work are given in Section 7.

2. Literature review

University timetabling problems have been extensively studied to provide institutions with efficient solving approaches. Institutions organize their teaching in different ways and have different hard constraints. Various soft constraints are used to assess solutions. Surveys and annotated bibliographies by Schaerf (1999), Burke and Petrovic (2002), Lewis (2008) Qu et al. (2009), Pillay (2014) and Babaei et al. (2015) give a good overall picture of how university timetabling problems have been addressed and they review different solving methods.

To solve timetabling problems, different types of approaches have been used. Approximate approaches such as heuristics (Abdullah, 2006), metaheuristics (Kaur and Saini, 2020) and hyperheuristics (Pillay, 2016), among others, are widely used. They provide a good trade-off between the quality of the solution and the computation time. Exact approaches have also been used for modelling purposes (McCollum et al., 2010), to provide lower bounds (Arbaoui et al., 2013, 2019) or to address certain problems (Arbaoui et al., 2015; Woumans et al., 2016). Exact approaches have sometimes been incorporated into approximate solution approaches such as matheuristics (Méndez-Díaz et al., 2016). Constraint-based approaches have also been used for university timetabling problems (Müller and Murray, 2010; Hoshino and Fabris, 2020).

The Student Scheduling Problems (also denoted as Student Sectioning Problems) that we consider here are timetabling problems that involve building individual timetables for every student by assigning students to sections for the different learning activities that constitute courses.

These SSPs can be considered as intrinsic to course timetabling problems or as standalone problems to be solved independently after timetabling. This reflects two different ways of organizing teaching. In the first, students choose their courses, and then given the students' choices, learning activities of courses are scheduled and students simultaneously assigned. SSP is therefore an implicit part of the scheduling operation. In the second, course timetables are first built according to historical data (previous semester's timetables are usually used and modified). Students then choose their courses, taking into account these proposed timetables. Once their choices have been made, students are assigned to the corresponding learning activities. In this case, SSP is a standalone problem.

When SSP is implicit in a university course timetabling problem, we face a type of problem usually denoted as Post Enrollment Course Timetabling (PE-CTT) as presented in International Timetabling Competitions (McCollum et al., 2007; Müller et al., 2018). This type of organization, in which students are assigned as timetables are created can also be encountered in high schools. A number of works that tackle PE-CTT-like problems can be found in the literature. Song et al. (2018) solved PE-CTT using an Iterated Local Search (ILS) approach. Akkan and Gülcü (2018) proposed a bi-criteria hybrid genetic algorithm with a robustness objective. In Hossain et al. (2019), the authors also considered the robustness for course timetabling problems. They proposed a PSO-based approach including swap sequences, selective search and forceful mechanisms for velocity operations. Müller and Murray (2010) also used an Iterative Local Search approach (ILS) to solve the problem at Purdue University. Thepphakorn and Pongcharoen (2020) proposed a hybrid cuckoo search approach to solve the course timetabling at the Faculty of Engineering, Naresuan University. Hoshino and Fabris (2020) used an ILP-based model to solve the PE-CTT at a Canadian high school. All these works consider SSP as a subproblem with variants depending on the institution's requirements. The most widely encountered requirement in student assignment is that a student should have a conflict-free timetable, but this is generally a secondary criterion (soft constraint) alongside the primary criteria that relate to the course/room/timeslot allocation that must be respected (hard constraints).

When SSP is solved as a standalone problem, the objective is to provide every student with an individual timetable, while also considering various requirements related to students and the institution. Laporte and Desroches (1986) provided a mathematical formulation to model their student scheduling problem, together with a heuristic for tackling the problem. Respecting student choices is the only hard constraint while time slot conflicts, room capacity and minimizing the number of moves between the sites are soft constraints. Sabin and Winter (1986) looked at the impact of a greedy approach to help reduce the cost of rising enrollment numbers. Miyaji et al. (1988) applied a goal programming approach to partition students into groups, taking into account orders of preference expressed by students and constraints relating to laboratories. Feldman and Golumbic (1990) proposed a Constraint-Satisfaction-Problem based approach, with the system being made available to students to draw up their timetables during the registration period. Tripathy (1992) proposed an Integer Linear Programming (ILP) approach to design a computer-aided system using students' choices of elective courses. Cheng et al. (2003) considered the SSP in North American high schools and proposed a multi-commodity flow formulation to satisfy students' choices. Van den Broek et al. (2009) considered the SSP of TU Eindhoven with additional constraints to be satisfied: a minimum number of students for each course, urgent courses and student workload. The authors solved TU Eindhoven's SSP using an ILP approach. Dostert et al. (2016) investigated the complexity analysis related to SSP, the authors explored the boundaries between easy and difficult cases.

Existing works on SSP addressed real-world problems and complexity analyses. We report here the constraints that have been considered in these works:

C1: Conflict-free, a student cannot attend two courses are the same time.

C2: Preferences, students' preferences should be satisfied.

C3: Capacity, a course/section capacity cannot be exceeded.

C4: Minimum number of students should be assigned to a course/section.

C5: Maximum student workload should not be exceeded per student.

C6: Course pre-assignment, students should be assigned to selected courses.

C7: Fixed/Non-fixed time slots, students have preferences for time slots.

C8: Course balancing, the sizes of course sections should be balanced.

C9: Moves between sites, certain moves between sites should be avoided.

C10: Urgent courses, some courses should be priority assigned to students.

C11: Spreading, courses should be spread over day (or week) for students.

C12: Breaks, students should have breaks between courses.

C13: Unavailability, students are unavailable for some time slots.

C14: Grouping of students, certain populations of students should be grouped.

	Type	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
Laporte and Desroches (1986)	R	S	H	S		S			S	S		S	S		
Sabin and Winter (1986)	R	H													
Miyaji et al. (1988)	R	H	S	H											
Feldman and Golumbic (1989)	R	H				H	H	S							
Tripathy (1992)	R	H	H	H											
Cheng et al. (2003)	R/C	H	H	H											
Van den Broek et al. (2009)	R/C	H	H	H	S	H	H				S				
Müller and Murray (2010)	R	S	H	H											
Dostert et al. (2016)	C	H	H	H										H	
Our study	R	H	H	H						S					S

Table 1: A comparison between the studied problem and existing works.

In Table 1 we show the sets of constraints considered. In column "Type", **R** stands for "Real-world" problem and **C** stands for "Complexity analysis". The other columns refer to the constraints C1 to C14, 'H' stands for "Hard" and 'S' stands for "Soft", this refers to whether the constraint has been considered as a hard or a soft constraint. As it can be seen, the constraints are considered either as hard or soft ones according to the university's requirement. The constraints C1 to C3 are widely encountered since they reflect usual requirements of many universities. The others are different since they reflect specific requirements. Comparisons, if any, are solely conducted with the systems used thus far by universities. The instances used for the experiments are not publicly available. To the best of our knowledge, no benchmarks on standalone SSP are available for comparison purposes.

Here, we consider a standalone SSP that is an NP-complete problem even when trying to find a schedule for one student as presented by (Cheng et al., 2003) in relation to their problem. Van den Broek et al. (2009) and Dostert et al. (2016) showed that the student scheduling problem becomes an NP-complete problem when sections have multiple events in the timetable and when no timeslot conflicts are allowed for students. These subproblems can also be identified for the UTC problem thus making the problem we face NP-hard.

The last line on Table 1 shows the constraints we consider in this study. The first three are considered in most works. In contrast to Laporte and Desroches (1986), we take into account the time needed for the moves by students. This permits to pay particular attention to students with reduced mobility. Moreover, we consider here the new constraint C14, the grouping of particular populations of students, with a twofold objective: i) furthering UTC's inclusion strategy for disabled students by pairing a student who needs particular assistance with students that can help, and ii) reducing our operating costs via an adapted sharing of existing UVs.

The practical SSP at UTC requires a specific combination of hard and soft constraints that has no been considered in the literature. The university's current requirements are detailled in the problem description section.

3. Problem description

We now outline the supported process and the heuristic used so far by our university. We first describe the student scheduling problem that our institution faces today. We then discuss data and parameters.

Overview of the supported process

Teaching at Université de Technologie de Compiègne (UTC) is provided by different academic departments. Every semester, each department chooses the UVs that will be offered to students. Sets of UVs that can be chosen by students enrolled in the department's program are identified. Timetables are then drawn up and made available to students through a web page that remains open for a few weeks.

Students select their UVs via an interactive web page, which prevents them selecting any set of UVs for which there is no feasible timetable. Typically a student will enroll on between five and seven different courses per semester. A student's choice of UVs for the current semester also needs to be checked against UVs taken in previous semesters, to ensure that current choices are consistent with previous choices within a long-term program of study. At the end of the registration period, the various heads of department monitor the students' choices. If a student's choice of UVs is deemed problematic, the head of department managing the curriculum urges the student to make changes.

Once enrollments have been finalized, there remains a large student scheduling problem to be solved in order to provide every student with an individual timetable. With thousands of students to be assigned to sections, manual assignment is out of the question.

The heuristic used so far

A greedy heuristic, denoted here as $\mathcal{H}$, was created some decades ago to assign students to sections for every activity of their selected UVs (Vayssade (1978)). This heuristic (1) determines an order in which students are to be processed and (2) fills up sections progressively. The rule used to assign students is *first fit*.

For a student, a rough estimate of the number of possible individual timetables is made, corresponding to the product of the number of available sections in the different activities constituting the student's chosen UVs. Students are then sorted in ascending order of this estimate, to be assigned to sections in turn. The reasoning behind the heuristic is that students with fewer potential individual timetables need to be assigned first, because these students are *a priori* more difficult to assign.

Assigning students is then done in several stages, during each of which a given percentage of any section's maximum capacity may not be exceeded. For instance, a three-stage assignment process could use the following percentages: 50%, then 70%, then 100%. At each stage students are processed in the established order, and are considered assigned once they have been assigned to a section for each activity of all their chosen UVs. Non-assigned students remain candidates at the following step, when additional capacity will be available. Sections are always considered in the same order. The progressive expansion of available capacity is intended to prevent sections that are examined earlier from filling up too quickly, given that the *first fit* rule will assign a student to the first feasible section.

Tuning the number of stages and the percentages of capacity to be used in $\mathcal{H}$, with the objective of reducing the number of students who remain non-assigned after the final stage, is a challenging task. Whatever the tuning, in practice there are always students who remain non-assigned, and it is not known whether a feasible solution exists that might potentially have avoided this situation.

Another limitation of this heuristic is that it cannot deal with learning activities considered as separate entities, but only with the complete set of learning activities for a given UV. To enable the heuristic to deal with students who are given exemptions from certain activities, administrative staff have had to devise workarounds, creating additional dummy UVs including only a subset of the learning activities of the original UV. This extra work is unwelcome, time-consuming and frequently a source of errors.

The university's current requirements

When students retake a UV, it is not always necessary to have them redo a learning activity that they previously did satisfactorily. The university is thus able to save money by reducing the number of lab sections and expenditure on costly consumables. The lecturer in charge of the UV decides on exemptions when students retake. The problem we face today is to assign every student to a set of learning activities, rather than to a set of UVs.

A student must be assigned to a section in all of his/her learning activities for the current semester. Where a complete assignment of the student fails,

he/she is said to be *non-assigned*. The assignment of a student to only a subset of his/her desired activities is not tolerated. This is a strict institutional requirement, showing why we need to compute better solutions than those provided by the heuristic $\mathcal{H}$. Our first criterion is therefore to minimize the number of *non-assigned students*.

Sections have events held on different sites, but moves between sites are undesirable where there is little time available to travel from one site to another.

Students with reduced mobility face additional difficulties, not only when moving from one site to another, but also when moving between buildings on the same site. For these students we seek to minimize moves that cannot be made comfortably because of the short time available. As part of our university's strategy for inclusion, our second and third criteria are therefore to minimize, for students with reduced mobility, the number of moves between buildings on the same site and the number of moves between sites when there is little time available (*hurried moves*).

A new requirement of our university is to group some populations of students, for certain learning activities, into as few sections as possible. Some students with disabilities are accompanied by other student volunteers who assist them in taking notes and/or in handling equipment and materials. There is consequently a need to specify sets of students, with the objective of assigning them to the same section. To meet the growing demand for engineering degrees, UTC allows some students to pursue their studies while simultaneously being on a long-term contract with a company. Tripartite *apprenticeship* agreements are drawn up between the student, the company and the university. Apprenticeship students alternate between periods of work inside the company and periods of study at UTC. The duration of periods is between 4 and 6 weeks. Today, because of the constraints of their particular timetables, apprenticeship students have specific *apprenticeship* learning activities, but many of these activities are indistinguishable from those undertaken by students who are at UTC full-time. If apprenticeship students are able to take part in standard learning activities rather than having their own special learning activities, teaching costs can be reduced. But to facilitate this, the apprenticeship students need to be assigned to the same sections. Grouping apprenticeship students into as few sections as possible of learning activities makes it easier to monitor their progress and to provide them with specific support (mini-group work), while they undertake the same activities as full-time students. Our fourth criterion is therefore to minimize the number of sections used in assigning certain sets of students to certain activities. This is a key element in allowing apprenticeship students and full-time students to have a shared curriculum.

UTC has recently decided that travel between sites by students when little time is available is undesirable for any student, irrespective of whether he/she has reduced mobility. Our fifth criterion is therefore to minimize the number of moves between sites where little time is available (*hurried moves*) for students in general.

Given the changing context, the order of priority on the criteria to be addressed is dictated by our university:

1. Minimizing the number of *non-assigned* students.
2. Minimizing the number of hurried moves for students with reduced mobility (sites, buildings).
3. Minimizing the number of sections used in assigning certain sets of students to certain activities.
4. Minimizing the number of hurried moves for students in general.

Data and parameters

The data and parameters are the following:

$\mathcal{S}$: set of students, size $n_{\mathcal{S}}$
$\mathcal{S}_{srm}$: set of **s**tudents with **r**educed **m**obility who have to move between sites and buildings, size $n_{\mathcal{S}_{srm}}$
$\mathcal{S}_{al}$: sets of students to be grouped into the minimum number of sections of an activity a, size $n_{\mathcal{S}_{al}}$
$\mathcal{S}_{osm}$: set of **o**ther **s**tudents who have to **m**ove between sites, size $n_{\mathcal{S}_{osm}}$
$\mathcal{A}$: set of learning activities, size $n_{\mathcal{A}}$
$\mathcal{S}_a$: set of students to be assigned into activity a, size $n_{\mathcal{S}_a}$
$\mathcal{K}$: set of sections, size $n_{\mathcal{K}}$
$\mathcal{K}_a$: set of sections of activity a, size $n_{\mathcal{K}_a}$
p_k : seating capacity of section k
$\mathcal{A}_s$: set of learning activities of student s, size $n_{\mathcal{A}_s}$
$\mathcal{K}_s$: set of sections ($\cup_{a \in \mathcal{A}_s} \mathcal{K}_{sa}$) where student s can be assigned to, size $n_{\mathcal{K}_s}$
$\mathcal{K}_{sa}$: set of sections of activity a where student s can be assigned to, size $n_{\mathcal{K}_{sa}}$
M : conflict matrix between sections, size $n_{\mathcal{K}} \times n_{\mathcal{K}}$
$G_s(\mathcal{K}_s, E_s)$: conflict graph of student s, one conflict graph for every student
Δt : parameters, times to move, different values for $\mathcal{S}_{srm}$ and $\mathcal{S}_{osm}$ students
M_{Δ_t} : matrices, used to check moves between sections with time to move $\leq \Delta t$
$\mathcal{D}_s, \mathcal{B}_s$: for a student $s \in \mathcal{S}_{srm}$, sets of quadruplets $\{a, k, a', k'\}$, used for checking hurried moves
$\mathcal{W}_s$: for a student $s \in \mathcal{S}_{osm}$, set of quadruplets $\{a, k, a', k'\}$, used for checking hurried moves

The set of students $\mathcal{S}$ to be assigned is given. The set of available sections for every student is known, and within these sections, the site and the building are known for every event where a room has been scheduled. We can compute $\mathcal{S}_{srm}$ and $\mathcal{S}_{osm}$, the set of **s**tudents with **r**educed **m**obility and the set of other students who may need to move between sites (*osm* stands for **o**ther **s**tudents' **m**oves). The sets $\mathcal{S}_{al}$ are given, corresponding to the sets of students to be grouped. The index l is required, since there may be more than one set of students to be grouped for a given activity.

The set of learning activities is $\mathcal{A}$, and the set of students to be assigned into activity a is $\mathcal{S}_a$. The set of sections planned in the timetables is $\mathcal{K}$, and every section k has a seating capacity p_k.

For a student s, $\mathcal{K}_s$ is the set of sections to be considered and $\mathcal{K}_{sa}$ is the set of sections into which this student can be assigned for an activity a.

M is the conflict matrix between sections, with $m_{kk'} = 1$ if sections k and k' conflict, 0 otherwise. Students cannot be assigned to two sections whose events overlap in time; sections of different activities with overlapping events are in conflict. For a student s, the conflict graph is denoted as $G_s(\mathcal{K}_s, E_s)$, where the nodes are sections $k \in \mathcal{K}_s$, and where there is an edge $[k, k'] \in E_s$ if $m_{kk'} = 1$.

The parameters Δt are used to set the time available for travel from one event to another. When timetables are drawn up, where possible a fifteen-minute break is scheduled between back-to-back events. This is often not enough to allow students with reduced mobility to move from one building to another on the same site or to move to a different site. For students in general, fifteen minutes may not be enough to move comfortably between sites. For **s**tudents with **r**educed **m**obility, our university has stipulated that the Δt sufficient for moves between buildings on the same site should be 30 minutes, the Δt for moves between sites 60 minutes. For other students, the Δt for moves between sites is 45 minutes.

The M_{Δ_t} matrices are to be used to detect whether there is enough time to move between events belonging to two non-conflicting sections. For two non-conflicting sections k, k' we update the time slots of their events, bringing forward the start and delaying the end by a value Δt in each case. Events may be selected that are planned on different sites or in different buildings on the same site. By checking intersections of updated events we build the M_{Δ_t} matrices. An entry is set to zero if there is strictly more than Δt time to move between the events of sections k and k', one otherwise.

For every student $s \in \mathcal{S}_{srm}$ we compute two sets of quadruplets $\mathcal{D}_s$ and $\mathcal{B}_s$ using the M_{Δ_t} matrices, and the sets $\mathcal{A}_s$ and $\mathcal{K}_{sa}$. If section k of activity a and section k' of activity a' are such that their events do not respect a given time Δt we set up a quadruplet $\{a, k, a', k'\}$. If student s is assigned to section k and section k', he/she does not have enough time to move comfortably. The first set $\mathcal{D}_s$ is for moves between sites and the second set $\mathcal{B}_s$ is for moves between buildings on the same site. For every student $s \in \mathcal{S}_{osm}$ we similarly compute a set $\mathcal{W}_s$ of quadruplets $\{a, k, a', k'\}$ for moves between sites.

4. rss preprocessing, lower bounds and conflict graph

To reduce the size of instances, we propose a procedure for reducing the sets of sections to be considered for assigning students. We propose a procedure for computing lower bounds for hurried moves (i.e., moves that cannot be accomplished in Δ_t time), and a lower bound on the number of sections used for grouping students together. Thus, the solutions that we obtain can be compared against the lower bound values. We look at the $G_s(\mathcal{K}_s, E_s)$ graphs and we highlight the importance of interval graphs for the clique formulation of the conflict constraints that we propose.

Preprocessing for reducing the set of sections for activities

We propose a preprocessing that may reduce the number of sections that can be used to assign a student s. Since a student cannot be assigned to more than

one section at the same time, the idea is to remove the sections that cannot be used from the initial set of sections $\mathcal{K}_s$ among which student s could be assigned.

For a student s and a learning activity a, any section in $\mathcal{K}_a$ may potentially be used when assigning s to a section k. Thus, initially $\mathcal{K}_{sa} = \mathcal{K}_a$. However, let a, a' be two activities for a student s, and let us suppose that activity a has just one section k. If there exists a section $k' \in \mathcal{K}_{sa'}$ such that $m_{kk'} = 1$, section k' cannot be used. We therefore have $\mathcal{K}_{sa'} \leftarrow \mathcal{K}_{sa'} - \{k'\}$, because section k' cannot be used in assigning student s to activity a', since student s must be assigned to the uniquely available section k of activity a. We may have $n_{\mathcal{K}_{sa}} \leq n_{\mathcal{K}_a}$. In that case, $\mathcal{K}_{sa'}$ is reduced, possibly now comprising in its turn just one section. The reduction continues to be applied until all uniquely available sections (whether they were single sections at the start or revealed subsequently) have been used.

We denote this preprocessing as **rss** (**r**educing **s**et of **s**ections). For a section k, checking whether or not $m_{kk'} = 1$ for all $k' \in (\mathcal{K}_s - \{k\})$ costs $\mathcal{O}(n_{\mathcal{K}_s})$ considering the initial set $\mathcal{K}_s$, so the overall complexity of **rss** for a student s is $\mathcal{O}((n_{\mathcal{K}_s})^2)$. It can also reduce the size of the $G_s(\mathcal{K}_s, E_s)$ conflict graph corresponding to student s.

Applying **rss** for every student s can reduce the overall size of an instance.

```
1  Procedure: MinMovesStudent(M,M_Δt,n_As,V_Ksa,V_T,n_moves)
   Input: M,M_Δt,n_As,V_Ksa,V_T
   Output: n_moves
2  if n_moves ≠ 0 then
3  |  K_sa ← V_Ksa[1 + sizeof(V_T)]
4  |  forall k ∈ K_sa do
5  |  |  if NoConflict(M,V_T,k) = true then
6  |  |  |  push_back(V_T, k)
7  |  |  |  if n_As ≠ sizeof(V_T) then
8  |  |  |  |  MinMovesStudent(M,M_Δt,n_As,V_Ksa,V_T,n_moves)
9  |  |  |  else
10 |  |  |  |  n_moves ← Min(CountHurriedMoves(M_Δt,V_T),n_moves)
11 |  |  |  end
12 |  end
```

Lower bounds for hurried moves

Given a M_{Δ_t} matrix, the *MinMovesStudent* procedure computes the smallest number of moves with an insufficient Δ_t for a student s to travel between two sites (or two buildings). It is a recursive tree search that builds all the individual timetables for a student s (without considering the other students) and assesses each of them in relation to a Δ_t time value and a geographic criterion (building or site). By summing these numbers for a set of students, we obtain a lower bound that can be used in assessing the quality of computed solutions.

The minimum number of moves is the global variable n_{moves}, set initially to $+\infty$. The M_{Δ_t} matrix is provided relative to the chosen criterion.

Vector $V_{\mathcal{K}_{sa}}$ contains the sets $\mathcal{K}_{sa}$ (sections belonging to activity a that can be considered in assigning student s), sorted in increasing order of size $n_{\mathcal{K}_{sa}}$. Vector V_T is a timetable under construction (conflict-free sections), initially empty.

We select the next set of sections to be considered for the next activity (line 3). Sections in the set $\mathcal{K}_{sa}$ are examined in turn (line 4). Given a node in the tree structure, we have one branch for every possible way of choosing a section k for activity a. Function *NoConflict*(M,V_T,k) checks whether section k is conflict-free with a timetable under construction V_T (line 5). If k is conflict-free it is added to V_T (line 6). For incomplete timetables (line 7, $n_{\mathcal{A}_s} \neq sizeof(V_T)$), the next activity is to be considered (line 8, recursive call). When a feasible timetable is encountered (line 9, $n_{\mathcal{A}_s} = sizeof(V_T)$), function *CountHurried-Moves*(M_{Δ_t},V_T) counts the number of moves, and n_{moves} is updated (line 10). Note that the procedure terminates as soon as a feasible timetable is found in which all moves comply with the stipulated Δ_t (line 2, $0 \neq n_{moves}$). The complexity of the *MinMovesStudent* procedure is $\mathcal{O}(\Pi_{a \in A_s} n_{K_{sa}})$.

By applying this procedure for every student in a given population ($\mathcal{S}_{srm}$ or $\mathcal{S}_{osm}$) we obtain an overall lower bound by totalling the computed n_{moves}. For **s**tudents with **r**educed **m**obility, we denote as $LB_{Si(srm)}$ and $LB_{Bu(srm)}$ the lower bounds for moves between **si**tes and **bu**ildings respectively. For **o**ther **s**tudents' **m**oves, we denote as $LB_{Si(osm)}$ the lower bound for moves between **si**tes.

Lower bound on numbers of sections for grouping sets of students together

Let $\mathcal{S}_{al}$ be a set (with size n_{Sal}) of students to be assigned into the smallest number of sections for an activity a. Consider all the sections $k \in \mathcal{K}_a$ for an activity a and their various capacities p_k. Given a set of students $\mathcal{S}_{al}$ for an activity a, the smallest number of sections that can be used is:

$$LB_{al} = \left\lceil \frac{n_{\mathcal{S}_{al}}}{Max_{k \in \mathcal{K}_a} p_k} \right\rceil$$

since capacities p_k may differ between sections. We obtain an overall lower bound LB_{gss} (gss standing for **g**rouping **s**ets of **s**tudents) by totalling the LB_{al} values.

$\mathbf{G_s}(\mathcal{K}_\mathbf{s}, \mathbf{E_s})$ graphs

For a student s, let us consider the set of maximal cliques $\mathcal{C}_s$ that are computed on the G_s graph. The set of maximal cliques covers all edges of the G_s graph. For every clique, conflicts between related sections can be prevented using a clique inequality. For every student, all conflicts can be prevented using clique inequalities. Moon and Moser (1965) proved that a graph may have an exponential number of cliques, and an ILP model with an exponential number of equations poses substantial problems.

However, some classes of graph have a linear number of maximal cliques as the number of nodes. This is the case for interval graphs (see Golumbic (2004)).

Unfortunately, although the G_s graphs are built using time intervals, they are not necessarily interval graphs, because in our problem a section may have more than one event.

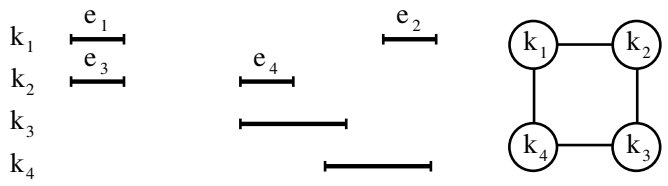

Figure 1: The section conflict graph built using time intervals is a C-4.

Figure 1 provides an illustrative example with four sections $\{k_1, k_2, k_3, k_4\}$ of different activities. Sections k_1 and k_2 each have two events. A student assigned to section k_1 must attend the two events (e_1, e_2) and a student assigned to section k_2 must attend the two events (e_3, e_4). Sections k_3 and k_4 have one event each. The graph built using time intervals is a C-4 and not an interval graph, as shown in Figure 1 (see Gilmore and Hoffman (2003) and Golumbic (2004)). This case may be embedded in any $G_s(\mathcal{K}_s, E_s)$.

Assuming that the case depicted in Figure 1 occurs only rarely, almost all G_s graphs will be interval graphs, since most sections have one event. If a $G_s(\mathcal{K}_s, E_s)$ graph is an interval graph there will be at most $n_{\mathcal{K}_s} = |\mathcal{K}_s|$ maximal cliques with equality if and only if the G_s graph has no edges (see Lekkeikerker and Boland (1962) and Shaohan and Wallis (1988)). So, assuming that almost all G_s graphs are interval graphs, using clique inequalities may give us fewer equations for preventing conflicts. If a G_s graph is not an interval graph, we likely have a reasonable number of maximal cliques, since the graph is small. We observed on our instances that G_s graphs had on average 46.4 nodes (sections per student).

A clique-based formulation with a reasonable number of equations can be used to impose the hard conflict constraints in a student assignment problem, provided that the number of non-interval graphs is small relative to the number of interval graphs. To ensure that this condition is met, we need to check whether G_s graphs are interval graphs. An interval graph has a perfect elimination ordering of its nodes (for each node k, k and the neighbors of k that occur after k in the order form a clique). In our experiments we check whether a perfect elimination ordering exists using the straightforward algorithm presented in Rose et al. (1976). The algorithm is based on a breadth-first search and finds a perfect elimination ordering, if any exists, in $O(|\mathcal{K}_s| + [E_s|)$ time.

5. Integer linear programming formulations

In this section we present our models for minimizing the five criteria detailed in Section 3. The instances that we are dealing with are sizeable. We investigate two formulations for the conflict constraints together with the clique formulation

that we propose. We first present our decision variables, before introducing an initial model that has a global objective function with weighted terms. We then present our lexicographic scheme for addressing the criteria to be optimized. We also present valid inequalities that may prove helpful in reducing computation time.

Decision variables

All data and parameters were introduced in Section 3 above. The primary boolean decision variables governing the assignment are the following:

$$T_s \begin{cases} 1 & \text{if student } s \in \mathcal{S} \text{ is assigned to a section for every activity } a \in \mathcal{A}_s, \\ 0 & \text{otherwise,} \end{cases}$$

$$Y_{sa} \begin{cases} 1 & \text{if student } s \in \mathcal{S} \text{ is assigned to a section for activity } a, \\ 0 & \text{otherwise,} \end{cases}$$

$$Z_{sak} \begin{cases} 1 & \text{if student } s \in \mathcal{S} \text{ is assigned to the section } k \text{ of activity } a, \\ 0 & \text{otherwise.} \end{cases}$$

The secondary boolean decision variables used to count the soft constraint violations are the following:

$$D_{saka'k'} \begin{cases} 1 & \text{if student } s \in \mathcal{S}_{srm} \text{ is assigned to section } k \text{ of activity } a \\ & \text{and to section } k' \text{ of activity } a', \text{ such that } \{a, k, a', k'\} \in \mathcal{D}_s \\ 0 & \text{otherwise,} \end{cases}$$

$$B_{saka'k'} \begin{cases} 1 & \text{if student } s \in \mathcal{S}_{srm} \text{ is assigned to section } k \text{ of activity } a \\ & \text{and to section } k' \text{ of activity } a', \text{ such that } \{a, k, a', k'\} \in \mathcal{B}_s \\ 0 & \text{otherwise,} \end{cases}$$

$$V_{k(al)} \begin{cases} 1 & \text{if at least one student } s \in \mathcal{S}_{al} \text{ is assigned to section } k \text{ of activity } a \\ 0 & \text{otherwise,} \end{cases}$$

$$W_{saka'k'} \begin{cases} 1 & \text{if student } s \in \mathcal{S}_{osm} \text{ is assigned to section } k \text{ of activity } a \\ & \text{and to section } k' \text{ of activity } a', \text{ such that } \{a, k, a', k'\} \in \mathcal{W}_s \\ 0 & \text{otherwise,} \end{cases}$$

For each student $s \in \mathcal{S}_{srm}$, for every quadruplet $\{a, k, a', k'\} \in \mathcal{D}_s$ we introduce a boolean variable $D_{saka'k'}$ and for every quadruplet $\{a, k, a', k'\} \in \mathcal{B}_s$ we introduce a boolean variable $B_{saka'k'}$. These variables are used to detect moves by students with reduced mobility where not enough time is available (*hurried moves*). M_1 corresponds to the number of $D_{saka'k'}$ variables, and M_2 the number of $B_{saka'k'}$ variables.

For each set $\mathcal{S}_{al}$ relating to activity a and for each $k \in \mathcal{K}_a$ we introduce a boolean variable $V_{k(al)}$. M_3 is the number of $V_{k(al)}$ variables.

For each student $s \in \mathcal{S}_{osm}$, for every quadruplet $\{a, k, a', k'\} \in \mathcal{W}_s$ we introduce a boolean variable $W_{saka'k'}$. These variables are used to detect hurried moves by other students. M_4 is the number of $W_{saka'k'}$ variables.

Integer linear programming model using weighted terms

We now present the ILP model for solving our assignment problems using a global objective function with weighted terms. We face sizeable instances, and here we look at three formulations for the hard conflict constraints, the idea being to retain the most compact formulation.

The objective function using weighted terms to address our assignment problems is the following:

Min

$$
\begin{aligned}
& M_1\, M_2\, M_3\, M_4 \underbrace{\left(n_S - \sum_{s \in \mathcal{S}} T_s \right)}_{N_{nas}} + M_2\, M_3\, M_4 \underbrace{\sum_{s \in \mathcal{S}_{srm}} \sum_{a,k,a',k' \in \mathcal{D}_s} D_{saka'k'}}_{N_{Si(srm)}} \\
& +\ M_3\, M_4 \underbrace{\sum_{s \in \mathcal{S}_{srm}} \sum_{a,k,a',k' \in \mathcal{B}_s} B_{saka'k'}}_{N_{Bu(srm)}} + M_4 \underbrace{\left(\sum_{\mathcal{S}_{al}} \sum_{k \in \mathcal{K}_a} V_{k(al)} \right)}_{N_{gss}} \\
& + \underbrace{\left(\sum_{s \in \mathcal{S}_{osm}} \sum_{\{a,k,a',k'\} \in \mathcal{W}_s} W_{saka'k'} \right)}_{N_{Si(osm)}}
\end{aligned}
\tag{1}
$$

subject to one conflict constraint formulation from among the following three:

$$
Z_{sak} + Z_{sa'k'} \leq 1 \quad \begin{cases} \forall s \in \mathcal{S} \quad \forall a, a' \in \mathcal{A}_s \\ \forall k \in \mathcal{K}_{sa} \quad \forall k' \in \mathcal{K}_{sa'} \\ \text{such that } m_{kk'} = 1 \end{cases} \tag{2a}
$$

$$
\sum_{\substack{a' \in \mathcal{A}_s \\ k' \in \mathcal{K}_{sa'} \\ k' \neq k}} m_{kk'} Z_{sa'k'} + \left(\sum_{\substack{a' \in \mathcal{A}_s \\ k' \in \mathcal{K}_{sa'} \\ k' \neq k}} m_{kk'} \right) Z_{sak} \leq \sum_{\substack{a' \in \mathcal{A}_s \\ k' \in \mathcal{K}_{sa'} \\ k' \neq k}} m_{kk'} \quad \begin{cases} \forall s \in \mathcal{S} \\ \forall a \in \mathcal{A}_s \\ \forall k \in \mathcal{K}_{sa} \end{cases} \tag{2b}
$$

$$
\sum_{k \in c} Z_{sak} \leq 1 \qquad \forall s \in \mathcal{S} \quad \forall c \in \mathcal{C}_s \tag{2c}
$$

and subject to:

$$
T_s \leq Y_{sa} \qquad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}_s \tag{3}
$$

$$
\sum_{k \in \mathcal{K}_{sa}} Z_{sak} = Y_{sa} \qquad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}_s \tag{4}
$$

$$\sum_{s \in \mathcal{S}} Z_{sak} \leq p_k \qquad \forall a \in \mathcal{A},\ \forall k \in \mathcal{K}_a \tag{5}$$

$$Z_{sak} + Z_{sa'k'} \leq 1 + D_{saka'k'} \qquad \forall s \in \mathcal{S}_{srm} \quad \forall \{a, k, a', k'\} \in \mathcal{D}_s \tag{6}$$

$$Z_{sak} + Z_{sa'k'} \leq 1 + B_{saka'k'} \qquad \forall s \in \mathcal{S}_{srm} \quad \forall \{a, k, a', k'\} \in \mathcal{B}_s \tag{7}$$

$$\sum_{s \in \mathcal{S}_{al}} Z_{sak} \leq min\left(p_k, n_{\mathcal{S}_{al}}\right)\ V_{k(al)} \qquad \forall a \in \mathcal{A} \quad \forall \mathcal{S}_{al} \quad \forall k \in \mathcal{K}_a \tag{8}$$

$$Z_{sak} + Z_{sa'k'} \leq 1 + W_{saka'k'} \quad \forall s \in \mathcal{S}_{osm} \quad \forall \{a, k, a', k'\} \in \mathcal{W}_s \tag{9}$$

$$T_s, Y_{sa}, Z_{sak}, D_{saka'k'}, B_{saka'k'}, V_{k(al)}, W_{saka'k'} \in \{0, 1\} \tag{10}$$

The weights used in the different terms of the objective function (1) are related to the order of priority dictated by our university. They were determined with a view to satisfying this order. The first term of the objective function is intended to minimize the **N**umber of **n**on-**a**ssigned **s**tudents N_{nas}. This term is weighted by $M_1\, M_2\, M_3\, M_4$ since our university wants first to minimize N_{nas}. The same rationale applies to the other terms included for the purposes of satisfying institutional priorities. The second term seeks to minimize $N_{Si(srm)}$, the number of hurried moves by **s**tudents with **r**educed **m**obility between *sites*, while the third term concerns $N_{Bu(srm)}$, their hurried moves between *buildings*. The fourth term seeks to minimize N_{gss}, the number of sections used in the assignment of certain sets of students to certain activities (**g**rouping **s**ets of **s**tudents). The fifth term seeks to minimize $N_{Si(osm)}$, the number of hurried moves between sites for other students (**o**ther **s**tudents' **m**oves).

Equations (2a), (2b) and (2c) ensure that, at each time slot, students are assigned to one section only. One formulation among these three is required to enforce the conflict constraints.

Equation (2a) ensures that no two conflicting sections ($m_{kk'} = 1$) are used at the same time for assigning student s. For a student s, there are as many equations as edges E_s. We refer to Equation (2a) as the *edge* formulation.

Equation (2b) ensures that a student s can be assigned to section k if not assigned to any sections that are in conflict with k. For a student s, the value $\sum m_{kk'}$ where $a' \in \mathcal{A}_s, k' \in \mathcal{K}_{sa'}, k' \neq k$, is the number of sections in conflict with section k of activity a. This represents the number of neighbors of section k in the $G_s(\mathcal{K}_s, E_s)$ graph. For a student s, there are as many equations as nodes $\mathcal{K}_s$. We refer to Equation (2b) as the *node* formulation.

Equation (2c) is a clique inequality. For a student s, we compute the set of maximal cliques $\mathcal{C}_s$ in $G_s(\mathcal{K}_s, E_s)$. The size of this set is $n_{\mathcal{C}_s}$. For a student s, the set of maximal cliques $\mathcal{C}_s$ covers all the edges of the graph, and there are as

many equations as maximal cliques in $G_s(\mathcal{K}_s, E_s)$. The conflict constraints are enforced by Equation (2c). We refer to Equation (2c) as the *clique* formulation.

Equation (3) links variables T_s and Y_{sa}. Equation (4) links variables Y_{sa} and Z_{sak} and ensures that a student is assigned to at most one section of activity a. Sections' capacity constraints are enforced by Equation (5).

Equations (6) and (7) set $D_{saka'k'} = 1$ or $B_{saka'k'} = 1$ if students with reduced mobility have hurried moves between sites or between buildings on the same site. Equation (8) sets $V_{k(al)} = 1$ if a student $s \in \mathcal{S}_{al}$ is assigned to section k of activity a. There are at most $min\,(p_k, n_{\mathcal{S}_{al}})$ students $s \in S_{al}$ assigned to section k. Equation (9) sets $W_{saka'k'} = 1$ if there is a hurried move between sites for a student $s \in \mathcal{S}_{osm}$ (other students).

The ILP model using weighted terms is denoted as $\mathcal{M}$ and consists of Equation (1), Equation (2a) or (2b) or (2c), and, Equations (3)-(10).

Lexicographic scheme

The ILP model with a global objective function and weighted terms using large M values may face difficulty to solve sizeable instances. The order of priority dictated by our institution can be advantageously enforced through a lexicographic optimization scheme. We look at a scheme with four stages.

At each stage, the best objective function value from the previous stage (obtained within a time limit) is maintained via a constraint, and the solution from the previous stage is given as an initial solution. The four problems we address in turn within the lexicographic scheme are the following:

1. Minimize the number of *non-assigned students* (N_{nas}).
2. Minimize the number of hurried moves between sites and between buildings for students with reduced mobility ($N_{Si(srm)}$ and $N_{Bu(srm)}$).
3. Minimize the number of sections used in assigning certain sets of students to certain activities (N_{gss}).
4. Minimize the number of hurried moves between sites for other students ($N_{Si(osm)}$).

The model that only minimizes N_{nas}, the unweighted first term of the objective function (1), is denoted as $\mathcal{M}_{nas}$. It consists of Equation (2a) or (2b) or (2c), Equations (3)-(5) and variables T_s, Y_{sa}, Z_{sak}.

The model that minimizes $N_{Si(srm)}$ and $N_{Bu(srm)}$ uses as an objective function:

$$M_2 \sum_{s \in \mathcal{S}_{srm}} \sum_{a,k,a',k' \in \mathcal{D}_s} D_{saka'k'} + \sum_{s \in \mathcal{S}_{srm}} \sum_{a,k,a',k' \in \mathcal{B}_s} B_{saka'k'}$$

The term $\sum_{s \in \mathcal{S}_{srm}} \sum_{a,k,a',k' \in \mathcal{D}_s} D_{saka'k'}$ is weighted by M_2, because for students with reduced mobility it is more important to minimize hurried moves between sites than between buildings. This model is denoted as $\mathcal{M}_{srm}$ and includes equations and variables relating to the two terms in addition to the equations comprising $\mathcal{M}_{nas}$, together with a constraint that maintains the N_{nas} value.

For the model that minimizes $N_{Si(osm)}$, the required additional equations and variables are added to $\mathcal{M}_{srm}$, together with two constraints that maintain the $N_{Si(srm)}$ and $N_{Bu(srm)}$ values. This formulation is denoted as $\mathcal{M}_{gss}$.

The model that minimizes N_{osm} is denoted as $\mathcal{M}_{osm}$ and consists of Equation (2a) or (2b) or (2c), and Equations (3)-(10), together with constraints that maintain the computed values at earlier stages.

Valid inequalities

The size of the problem increases with the number of criteria. We are therefore dealing with large SSP problems. Valid inequalities can be helpful in reducing computation time and improving results. We propose three valid inequalities.

Students enrolled in the same activities are assigned

Let us consider students enrolled in two activities a and a', who need to be assigned to a section $k \in \mathcal{K}_a$ and a section $k' \in \mathcal{K}_{a'}$. We propose the following as valid inequalities:

$$\left.\begin{array}{l} \forall a, a' \in \mathcal{A} \\ a \neq a' \\ \mathcal{S}_a \cap \mathcal{S}_{a'} \neq \emptyset \end{array}\right\} \qquad \sum_{s \in (\mathcal{S}_a \cap \mathcal{S}_{a'})} \sum_{k \in \mathcal{K}_a} Z_{sak} = \sum_{s \in (\mathcal{S}_a \cap \mathcal{S}_{a'})} \sum_{k' \in \mathcal{K}_{a'}} Z_{sa'k'} \qquad (11)$$

since students $s \in \mathcal{S}_a \cap \mathcal{S}_{a'}$ have to be assigned into both activities a and a'.

Tightening maximum numbers of students to be assigned to an activity

The number of students enrolled in activity a is n_{Sa} (the size of set $\mathcal{S}_a$). So the number of students assigned is also bounded by n_{Sa}. We propose:

$$\forall a \in \mathcal{A} \qquad \sum_{s \in \mathcal{S}_a} \sum_{k \in \mathcal{K}_a} Z_{sak} \leq n_{Sa} \qquad (12)$$

since the number of students assigned to sections $k \in \mathcal{K}_a$ cannot exceed n_{Sa}.

Student completely assigned or not

We recall that as long as there remains an activity for which a student has not been assigned to a section, that student is considered an *non-assigned student*. Partial assignments are precluded by a strict institutional constraint, which we use in creating a valid inequality.

Let us consider a student s and two different activities $a, a' \in \mathcal{A}_s$. Since student s has to be assigned into these two activities, we propose:

$$\left.\begin{array}{l} \forall s \in \mathcal{S} \\ \forall a, a' \in \mathcal{A}_s \\ a \neq a' \end{array}\right\} \qquad Y_{sa} = Y_{sa'} \qquad (13)$$

as valid inequalities. Hence, student s is assigned either into all his/her activities or (exclusive) into no activities.

6. Experiments and results

In our experiments, our objectives were: (i) to demonstrate the effectiveness of the *rss* preprocessing in reducing instance sizes and to compare the compactness of the three formulations of the conflict constraints; (ii) to compare performances between, on the one hand, the model with global objective function, and on the other, the models in our lexicographic scheme; (iii) to compare numbers of non-assigned students obtained using the heuristic used in the past by our university with those obtained using ILP models; (iv) to monitor the performance of valid inequalities as the number of non-assigned students increases; (v) to assess the quality of the solutions given by the different models in the lexicographic scheme with respect to lower bounds and to show to what extent initial solutions could be improved.

Tests were done using a CPLEX 12.10 IBM (2020) solver with a single thread and the *MipEmphasis* parameter set to *feasibility*, using C++ compiled with gcc version 7.5.0, on a machine with an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz and 64 GB of RAM.

Instance characteristics

The characteristics of the fifteen problem instances obtained from Université de Technologie de Compiègne (UTC) are shown in Table 2. Labels of instances in the first column correspond to **F**all and **S**pring semesters from spring 2013 to spring 2020. Data for previous semesters (before spring 2013) are not available. The COVID-19 pandemic has impacted the learning organisation of Fall 2020. Some constraints were added and others removed and the problem solved is different.

	n_{UV}	n_A	n_K	n_S	$n_{\mathcal{S}_{srm}}$	$n_{\mathcal{S}_{gss}}$	$n_{\mathcal{S}_{osm}}$
S13	311	609	1,500	2,396		11	1,833
F13	310	608	1,479	2,344		68	1,816
S14	303	606	1,425	2,275		14	1,546
F14	294	572	1,351	2,308	40		1,381
S15	203	604	1,399	2,351	40		1,367
F15	283	547	1,346	2,303	50	26	1,337
S16	297	590	1,355	2,316	54	5	1,321
F16	322	582	1,355	2,490	48		1,457
S17	317	607	1,344	2,363	48	6	1,304
F17	317	581	1,334	2,603	68		1,459
S18	328	626	1,340	2,412	71	82	1,254
F18	329	596	1,356	2,673	68		1,383
S19	325	630	1,329	2,688	81	153	1,432
F19	330	603	1,351	2,610	81		1,588
S20	338	652	1,371	2,606	78	141	1,501
Avg	307	601	1,376	2,449	60	56	1,465

Table 2: Instance characteristics.

Columns n_{UV}, n_A, and n_K correspond respectively to the number of UVs (courses), the total number of different learning activities (lectures, labs, etc.),

and the total number of sections for these learning activities.

Columns n_S, $n_{\mathcal{S}_{srm}}$, $n_{\mathcal{S}_{gss}}$, and $n_{\mathcal{S}_{osm}}$ concern numbers of students.

It can be seen that the number of enrolled students n_S has increased over the years. For reasons of confidentiality, the data that we were given for the instances in relation to students with reduced mobility were anonymized lists of students with disabilities, and contained no indication of the nature of the disability. For this reason, for the purposes of our tests, all students with disabilities were deemed to be students with reduced mobility. The column $n_{\mathcal{S}_{srm}}$ reports the number of disabled students, all considered to be students with reduced mobility.

Real data for the students that needed to be grouped together were available for the nine most recent instances. Some of these data correspond to disabled students who required help for certain learning activities (including handling equipment and note taking). When they were available, we also took into account data relating to apprenticeship students who could potentially be assigned to the same learning activities as full-time students rather than to specific activities for apprenticeship students. The column $n_{\mathcal{S}_{gss}}$ reports the number of students to be assigned into as few sections as possible for certain learning activities. For these instances, there are on average 56 apprenticeship students, but the more successful we are in grouping these students together into as few sections as possible, the greater the potential for reducing costs, and the greater the likelihood of hosting more apprenticeship students in the future.

The column $n_{\mathcal{S}_{osm}}$ shows that more than half of the students can be faced with moves between sites. This criterion is important, especially given that the number of sites is likely to increase in the future.

The size of these instances shows that these SSP problems cannot be processed manually. Moreover, the heuristic used in past years has become obsolete, since it fails to address the new requirements of our university.

Impact of the rss preprocessing on $G_s(\mathcal{K}_s, E_s)$ conflict graphs, and on the numbers of constraints for *Edge*, *Node* and *Clique* formulations

The impact of the **rss** preprocessing that we presented in Section 4 is shown in Table 3. The last row corresponds to the averages computed over all the instances.

For each instance, the columns $\overline{\mathcal{K}}_s$ show the average number of nodes in $G_s(\mathcal{K}_s, E_s)$ graphs, and the columns %IG show the percentage of these graphs that are interval graphs, respectively without preprocessing and with **rss** preprocessing.

It can be seen from Table 3 that the *rss* preprocessing reduces the number of nodes. When there is no *rss* preprocessing the average number of nodes (last row) is 46.4 and the average percentage of interval graphs is 97.7%. When *rss* preprocessing is applied the average number of nodes is 37.8 and the average percentage of interval graphs is 99.9%. The case illustrated in Figure (1) occurs only rarely, since there are fewer activities for which sections have multiple events than activities whose sections have one event only. By removing a section which has multiple events from a $\mathcal{K}_{sa}$ set for a student s, we reduce the size

	$G_s(\mathcal{K}_s, E_s)$ graphs				Number of conflict constraints in thousands					
	no rss		rss		no rss			rss		
	$\overline{\mathcal{K}}_s$	%IG	$\overline{\mathcal{K}}_s$	%IG	Edge	Node	Clique	Edge	Node	Clique
S13	51.3	96.3	40.3	99.9	837	119	47	480	82	**31**
F13	53.9	97.3	43.9	99.9	763	122	51	490	90	**36**
S14	47.7	96.7	37.3	100.0	634	104	43	361	71	**27**
F14	52.5	97.6	42.7	99.9	679	117	52	434	85	**36**
S15	48.1	97.3	37.1	100.0	701	109	45	388	72	**27**
F15	50.7	97.3	40.9	99.9	620	112	49	384	80	**34**
S16	46.9	98.5	36.6	100.0	649	104	44	364	70	**27**
F16	49.2	79.1	39.3	100.0	691	118	52	411	83	**35**
S17	46.1	97.7	35.8	99.9	638	103	43	350	69	**26**
F17	36.0	97.6	44.8	100.0	616	112	47	370	79	**32**
S18	42.7	98.9	33.5	100.0	589	98	40	330	65	**25**
F18	44.3	97.5	35.2	99.9	616	112	49	359	78	**32**
S19	39.4	98.6	30.5	100.0	554	100	41	290	64	**24**
F19	46.6	78.5	37.2	98.8	630	116	49	379	81	**33**
S20	41.2	98.5	32.2	99.9	535	101	41	293	66	**24**
Avg	46.4	97.7	37.8	99.9	650	110	46	379	76	**30**

Table 3: Average size of nodes of $G_s(\mathcal{K}_s, E_s)$ graphs and percentages of interval graphs, without and with **rss** preprocessing. Number of conflict constraints (in thousands) for the **Edge**, the **Node** and the **Clique** formulations, without and with **rss** preprocessing.

of $\mathcal{K}_s$ and are able to obtain an interval graph for $G_s(\mathcal{K}_s, E_s)$, although it was not an interval graph initially. The *rss* preprocessing also helps in reducing the number of non-interval graphs.

The right-hand part of Table 3 shows the number of conflict constraints for the *Edge*, *Node* and *Clique* formulations, first without, and then with *rss* preprocessing. For the sake of compactness, the values shown are in thousands.

It can be seen that *rss* preprocessing reduces the number of conflict constraints, irrespective of the formulation used. Moreover, the number of conflict constraints using the *Clique* formulation is consistently smaller than with the two others – it is smaller by one order of magnitude than with the *Edge* formulation (30 on average, as opposed to 379), and it is less than half of the number occurring with the *Node* formulation (30 as opposed to 76).

The computation time for applying **rss** preprocessing to every student in the set $\mathcal{S}$ is of the order of ten seconds per instance. Almost all $G_s(\mathcal{K}_s, E_s)$ are interval graphs, but a few are not. To compute sets of maximal cliques for all students, we use the approach proposed by Östergård (2002). This takes of the order of ten seconds for an instance.

The **rss** preprocessing can advantageously be used to reduce the size of SSP problem instances when a student cannot attend two events at the same time. All of the tests reported below were done with **rss** preprocessing.

$\mathcal{M}$ model using *Edge*, *Node*, and *Clique* formulations

In this section, we report on our tests using the $\mathcal{M}$ model, which has a global objective function with a weighted sum of terms. We tested the *Edge*, the *Node*,

and the *Clique* formulations of the conflict constraints.

	$\mathcal{M}$ model					
	Edge		Node		Clique	
	Status	t (s)	Status	t (s)	Status	t (s)
S13	O	56.4	O	98.4	O	52.0
F13	NoSol	-	NoSol	-	NoSol	-
S14	O	33.8	O	50.9	O	21.1
F14	NoSol	-	NoSol	-	NoSol	-
S15	O	30.0	O	57.9	O	30.0
F15	NoSol	-	NoSol	-	NoSol	-
S16	NoSol	-	NoSol	-	O	2289.4
F16	NoSol	-	NoSol	-	NoSol	-
S17	O	1802.7	NoSol	-	O	768.8
F17	O	832.9	NoSol	-	O	1575.7
S18	O	1936.0	NoSol	-	O	2654.8
F18	F	-	NoSol	-	NoSol	-
S19	NoSol	-	NoSol	-	O	2303.7
F19	O	834.4	NoSol	-	O	488.5
S20	F	-	NoSol	-	O	3303.3
Optimal	7/15		3/15		10/15	
Feasible	2/15		0/15		0/15	
No Solution	6/15		12/15		5/15	

Table 4: Status of solutions obtained with the $\mathcal{M}$ model using *Edge*, *Node*, and *Clique* formulations. Time limit 3600s.

A summary of the different solutions obtained within a time limit of 3600s is given in Table 4. For each instance the column *Status* contains either O for **O**ptimal, F for **F**easible (but not optimal), or NoSol for **No Sol**ution. Column t gives the computation time in seconds ("-" where the time limit was exceeded).

It will be remarked that for several of the instances, $\mathcal{M}$ fails to give a solution within the one-hour time limit, whatever formulation is used for the conflict constraints model. When optimality is attained using the $\mathcal{M}$ model with a global objective function, we obtain the same values for N_{nas}, $N_{Si(srm)}$, $N_{Bu(srm)}$, N_{gss}, $N_{Si(osm)}$ as those computed within the lexicographic scheme (see below). The *Edge* formulation attains optimality in seven out fifteen instances, the *Node* formulation attains optimality in three out fifteen instances, and the *Clique* formulation attains optimality in ten out fifteen instances.

The $\mathcal{M}$ model with the *Clique* formulation of conflict constraints is the most successful at providing optimal solutions. Table 3 shows that there are fewer equations with the *Clique* formulation, and this is the formulation of conflict constraints that we chose for the four models in our lexicographic scheme.

Comparison of the previously used heuristic and the $\mathcal{M}_{nas}$ model

Table 5 shows results given by the previously used heuristic $\mathcal{H}$ and shows results obtained by the first model $\mathcal{M}_{nas}$ in the lexicographic scheme. The columns N_{nas} and t show the number of non-assigned students and computation time t obtained using the heuristic $\mathcal{H}$ (the best results that university personnel were able to obtain), and using the $\mathcal{M}_{nas}$ model.

		$\mathcal{H}$		$\mathcal{M}_{nas}$	
	$n_{\mathcal{S}}$	N_{nas}	t (s)	N_{nas}	t (s)
S13	2,396	11	1.32	**0**	4.44
F13	2,344	21	2.76	**11**	15.89
S14	2,275	16	0.99	**0**	8.45
F14	2,308	23	5.30	**5**	4.89
S15	2,351	20	0.86	**0**	7.23
F15	2,303	13	1.05	**0**	9.32
S16	2,316	19	1.52	**4**	8.80
F16	2,490	25	1.90	**0**	12.39
S17	2,363	21	2.82	**0**	5.24
F17	2,603	20	2.49	**0**	4.82
S18	2,412	28	1.84	**0**	2.87
F18	2,673	48	4.65	**2**	14.71
S19	2,688	35	1.51	**0**	3.34
F19	2,610	62	4.26	**0**	7.41
S20	2,606	41	1.64	**0**	3.23
Avg	2,449	26.8	2.3	**1.5**	7.54

Table 5: Comparison of heuristic $\mathcal{H}$ and $\mathcal{M}_{nas}$ model.

Using the $\mathcal{M}_{nas}$ model, optimal results were obtained for all instances. In every case the computation time was less than twenty seconds. Computation times were shorter using the $\mathcal{H}$ heuristic, but solutions are not optimal. With the $\mathcal{M}_{nas}$ model all students were assigned in eleven out of the fifteen instances, and in the remaining four instances the number of non-assigned students was considerably reduced. The heuristic $\mathcal{H}$ systematically failed to achieve $N_{nas} = 0$.

In the past, every case of a non-assigned student had to be processed individually by hand, and attempting find an alternative feasible individual timetable was unwelcome and time-consuming. Staff performing this task were working "in the dark", not knowing whether a complete assignment into all of a student's activities was even possible. The analysis of results is possible only when optimality is attained, and it cannot be done in relation to the results obtained by the $\mathcal{H}$ heuristic.

The $\mathcal{M}_{nas}$ model not only achieved better results, but it also completely eliminated the extra work of managing dummy UVs, made necessary by the heuristic's inability to satisfy the current requirement of assigning students to a set of learning activities rather than to a set of UVs.

Effectiveness of valid inequalities

The number of students has increased over the years and will continue to grow, and the size of the problem increases with the number of criteria. We expect to have to address more difficult instances in the future. In Section 5 above we proposed valid inequalities that may be helpful in addressing harder instances. To test the valid inequalities, we decided to derive harder instances by reducing the capacity of all labs sections by one, in order to increase the number of non-assigned students.

Table 6 has two columns for each of the tests done on different derived instances. The first column is the number of non-assigned students N_{nas}, and the second is for the computation time t (shown as "-" where a time limit of

Derived instance	$\mathcal{M}_{nas}$		$\mathcal{M}_{nas}$+(11)		$\mathcal{M}_{nas}$+(12)		$\mathcal{M}_{nas}$+(13)	
	N_{nas}	t (s)	N_{nas}	t (s)	N_{nas}	t (s)	N_{nas}	t (s)
S13	52	-	2389	-	52	-	**51**	-
F13	91	-	2341	-	2274	-	**71**	-
S14	**35**	190.3	2266	-	**35**	-	**35**	**30.8**
F14	2227	-	2305	-	2227	-	**63**	-
S15	52	-	2341	-	52	-	**51**	**111.8**
F15	**63**	36.6	2289	-	**63**	44.3	**63**	**15.4**
S16	**48**	-	2313	-	**48**	-	**48**	-
F16	2389	-	2487	-	2308	-	**60**	-
S17	**50**	11.3	2313	-	**50**	13.2	**50**	**5.1**
F17	**53**	18.6	2595	-	**53**	17.3	**53**	**7.0**
S18	**49**	12.7	2397	-	**49**	15.0	**49**	**4.4**
F18	2409	-	2550	-	2409	-	**58**	-
S19	**54**	-	2526	-	**54**	29.2	**54**	**6.9**
F19	**56**	45.4	2588	-	**56**	50.8	**56**	**10.1**
S20	**55**	15.7	2599	-	**55**	18.3	**55**	**5.6**
Best Result	9/15		0/15		9/15		15/15	
Optimal		7/15		0/15		7/15		9/15
Best Time		0/15		0/15		0/15		9/15

Table 6: Formulation $\mathcal{M}_{nas}$: impact of valid inequalities on derived instances, time limit 300s.

300s was exceeded).

The heading $\mathcal{M}_{nas}$ corresponds to results without valid inequalities. In nine out of the fifteen instances the best results were obtained, and in seven of them optimality was proved within the time limit.

The heading "$\mathcal{M}_{nas}$ + (11)" corresponds to results for the "*students enrolled in the same activities are assigned*" valid inequality. Clearly this valid inequality is not satisfactory: the time limit was exceeded in all cases because of the large number of equations added, and no feasible solution was obtained.

The heading "$\mathcal{M}_{nas}$ + (12)", corresponds to results for the "*tightening maximum numbers of students to be assigned to an activity*" valid inequality. In nine out of the fifteen instances the best results were obtained, and in seven of them optimality was proved. This valid inequality does not appear to have a clear advantage over the $\mathcal{M}_{nas}$ model.

Best results were obtained for all instances using the "*student completely assigned or not*" valid inequality (the heading "$\mathcal{M}_{nas}$ + (13)"). Optimality was proved for nine out of the fifteen instances. Moreover, this valid inequality achieved the best computing time in nine out of the fifteen instances. This valid inequality is useful when used in conjunction with the $\mathcal{M}_{nas}$ model in order to address harder instances with greater numbers of students that cannot be assigned to all their learning activities.

Below, unless otherwise specified, the "*student completely assigned or not*" valid inequality is used.

Minimizing hurried moves for students with reduced mobility

In Section 5, we presented the $\mathcal{M}_{srm}$ model, which we propose as the second step in the lexicographic scheme for minimizing hurried moves (between sites and buildings) for students with reduced mobility. In Section 4, we presented the *MinMovesStudent* procedure that we use to compute $LB_{Si(srm)}$ and $LB_{Bu(srm)}$,

the lower bounds for undesired moves between sites and buildings.

Table 7 shows results obtained using the $\mathcal{M}_{srm}$ model for the twelve instances for which data are available regarding students with disabilities, all considered as having reduced mobility. For the purposes of comparison, these results are shown alongside the lower bounds $LB_{Si(srm)}$ and $LB_{Bu(srm)}$ and the initial values $N_{Si(srm)}$ and $N_{Bu(srm)}$ from the $\mathcal{M}_{nas}$ model.

		Lower Bounds		Initial Solution		$\mathcal{M}_{srm}$		
	$n_{\mathcal{S}_{srm}}$	$LB_{Si(srm)}$	$LB_{Bu(srm)}$	$N_{Si(srm)}$	$N_{Bu(srm)}$	$N_{Si(srm)}$	$N_{Bu(srm)}$	t (s)
F14	40	2	11	17	33	**2**	**8**	15.7
S15	40	3	1	9	21	**3**	**1**	6.2
F15	50	11	7	35	40	**11**	11	24.6
S16	54	12	7	26	43	**12**	12	4.2
F16	48	9	5	28	35	**9**	6	11.6
S17	48	10	8	28	30	**10**	10	3.7
F17	68	12	13	35	45	**12**	13	5.4
S18	71	32	15	71	61	33	20	2.6
F18	68	11	8	48	51	**11**	12	17.3
S19	81	20	13	54	45	21	16	4.4
F19	81	17	17	50	58	**17**	**17**	6.5
S20	78	24	6	57	37	**24**	9	3.2

Table 7: Formulation $\mathcal{M}_{srm}$ minimizing undesired moves for students with reduced mobility.

Column $n_{\mathcal{S}_{srm}}$ is the number of students, and columns $LB_{Si(srm)}$ and $LB_{Bu(srm)}$ are the lower bounds for moves between sites and buildings respectively that need to be done in less than Δt. "Initial Solution" shows the initial values $N_{Si(srm)}$ (for sites) and $N_{Bu(srm)}$ (for buildings) computed using the $\mathcal{M}_{nas}$ model. "$\mathcal{M}_{srm}$" shows the values obtained for $N_{Si(srm)}$ and $N_{Bu(srm)}$, along with the computation time in seconds.

Optimal results were obtained for all instances using the $\mathcal{M}_{srm}$ model while maintaining the previous optimized values of N_{nas}. It will be remarked that the initial solutions included considerably more hurried moves between sites and buildings. With $\mathcal{M}_{srm}$, in ten out of twelve instances we have $N_{Si(srm)} = LB_{Si(srm)}$, and in three out of the instances we also have $N_{Bu(srm)} = LB_{Bu(srm)}$ (shown in bold print in Table 7).

The $\mathcal{M}_{srm}$ model succeeded in minimizing hurried moves for students with reduced mobility, the number of moves are close to lower bounds, these students have less uncomfortable individual timetables. This feature of the lexicographic scheme is a welcome addition to our university's inclusion strategy.

Grouping sets of students

Institutions nowadays may seek to group some sets of students together. In Section 3, we mentioned some specific examples (students with disabilities and students on long-term placements in industry). In Section 5, we presented the $\mathcal{M}_{gss}$ model that we propose as the third step of the lexicographic scheme for minimizing the number of sections used to assign sets of students for activities. In Section 4, we presented the lower bound LB_{al} to be used for the purpose of comparison.

			Initial Solution	$\mathcal{M}_{gss}$	
	$n_{\mathcal{S}_{gss}}$	LB_{gss}	N_{gss}	N_{gss}	t (s)
S13	11	1	5	**1**	4.8
F13	68	5	25	**5**	146.7
S14	14	2	5	**2**	3.9
F15	26	1	3	2	87.8
S16	5	1	2	**1**	4.2
S17	6	2	3	**2**	3.7
S18	82	6	21	**6**	6.7
S19	153	12	40	14	4.9
S20	141	12	47	**12**	6.6

Table 8: Results of model $\mathcal{M}_{gss}$ for grouping sets of students.

Table 8 shows results obtained using the $\mathcal{M}_{gss}$ model for the nine instances for which data can be retrieved. We compare the results with the values of the LB_{gss} lower bound and with the initial values of N_{gss} given by the preceding model in the lexicographic scheme.

Column $n_{\mathcal{S}_{gss}}$ is the total number of students concerned by groupings, and column LB_{gss} is the computed lower bound. The "Initial Solution" column N_{gss} gives the number of sections used in assigning students. The two "$\mathcal{M}_{gss}$" columns are respectively the result given by our $\mathcal{M}_{gss}$ model and the computation time in seconds.

The $\mathcal{M}_{gss}$ model attains optimality for all instances while maintaining the previous optimized values. It can be seen that the N_{gss} values are an improvement on the initial values, with LB_{gss} attained in seven out of nine instances (in bold print in Table 8).

In the past, using the heuristic approach, administrators sought to manage the grouping of students by trial and error, booking seating capacity in order to assign students by hand to certain sections. It was tedious, time-consuming work, but with the heuristic approach they had no other choice. The grouping of sets of students was therefore discontinued except when strictly necessary for disabled students.

As Table 8 shows, the grouping of certain sets of students is managed painlessly using our proposed $\mathcal{M}_{gss}$ model. Students are successfully grouped into the smallest possible numbers of sections. This is useful for managing the individual assistance given to some disabled students in some learning activities. It can also reduce costs, allowing teaching in some cases to be tailored to the needs of particular subgroups within a shared overall curriculum.

Minimizing hurried moves by students in general

In Section 5, we presented the $\mathcal{M}_{osm}$ model that we propose as the fourth step in our lexicographic scheme for minimizing the number of hurried moves between sites. In Section 4, we presented the *MinMovesStudent* procedure to compute a lower bound on the number of these hurried moves for students as a whole, yielding the overall lower bound LB_{osm}.

			Initial Solution		$\mathcal{M}_{osm}$		$\mathcal{M}$	Lexicographic scheme, total
	n_{osm}	LB_{osm}	N_{osm}	N_{osm}	no (13) t (s)	with (13) t (s)	t (s)	t (s)
S13	1833	703	1840	**756**	13.6	13.6	52.0	**22.9**
F13	1816	483	1526	**526**	2865.0	2845.4	NoSol	**3007.9**
S14	1546	552	1400	**596**	7.6	7.6	21.1	**19.9**
F14	1381	320	1059	**320**	51.2	50.8	NoSol	**71.4**
S15	1367	465	1124	**490**	8.4	8.4	30.0	**21.8**
F15	1337	298	1026	**329**	74.5	76.7	NoSol	**198.4**
S16	1321	409	1000	**427**	18.9	19.6	2289.4	**36.8**
F16	1457	378	1184	**415**	59.8	59.8	NoSol	**83.8**
S17	1304	467	1116	**517**	8.1	8.2	768.7	**20.9**
F17	1459	351	1162	**384**	13.0	12.9	1575.6	**23.1**
S18	1254	444	1070	**497**	5.6	5.6	2654.7	**17.8**
F18	1383	286	1046	**346**	1346.8	1355.8	NoSol	**1387.8**
S19	1432	461	1242	**524**	11.0	10.5	2303.7	**23.2**
F19	1588	378	1269	**468**	12.6	12.6	488.5	**26.5**
S20	1501	570	1359	**628**	10.7	10.4	3303.2	**23.4**
Avg	1465	438	1228	**482**	300.5	299.9		

Table 9: Model $\mathcal{M}_{osm}$ minimizing hurried moves by students in general. Comparing the $\mathcal{M}$ model with a global objective function and the lexicographic scheme.

Table 9 shows the results from the $\mathcal{M}_{osm}$ model for the fifteen real instances alongside the LB_{osm} lower bound and the initial values given by the preceding model in the lexicographic scheme. Best results are in bold print. The last row gives the averages computed over all the instances.

Column n_{osm} is the number of students liable to move between sites (more than the half of all students – see column n_S in Table 2). Column LB_{osm} gives lower bounds, the numbers of unavoidable moves between sites where Δt is not respected (see Section 4).

The "Initial Solution" column N_{osm} gives the number of hurried moves before applying the $\mathcal{M}_{osm}$ model assessed on initial solutions, and immediately to the right of this, under the heading "$\mathcal{M}_{osm}$", we have the number of moves given by $\mathcal{M}_{osm}$.

$\mathcal{M}_{osm}$ is the final model in the lexicographic scheme and includes the whole set of equations. As previously stated, there are a large number of students liable to move between sites. We therefore tested the $\mathcal{M}_{osm}$ model both without and with the valid inequality equation (13) in order to observe its impact. The columns "no (13)" and "with (13)" give computation times t in seconds without and with the valid inequality Equation (13).

Optimal results were obtained for all instances using the $\mathcal{M}_{osm}$ model while maintaining the previous optimized values. A comparison of the two $\mathcal{M}_{osm}$ computation times shows that the valid inequality has no significant negative or positive impact on computing time overall. The real instances encountered so far have at most eleven students that cannot be assigned (see Table 5), which is not enough for any improvement to be observed. However, the number of students continues to rise and we will undoubtedly face instances with more

students that cannot be assigned. Given that we observed a beneficial impact using the valid inequality on a harder instance with around fifty non-assigned students (see Table 6), we have chosen to embed it to be able to address future harder instances.

It can be seen from Table 2 that initial solutions are substantially improved. On average, the number of hurried moves assessed in the initial solution is almost three times the lower bound, while optimal solutions given by the $\mathcal{M}_{osm}$ model are only ten percent above the lower bound.

$\mathcal{M}$ model versus lexicographic scheme ($\mathcal{M}_{nas}$, $\mathcal{M}_{srm}$, $\mathcal{M}_{gss}$, $\mathcal{M}_{osm}$)

The "$\mathcal{M}$" column in Table 9 gives the computation time for the $\mathcal{M}$ model with the global objective function that attains the greatest number of optimal solutions within a one-hour time limit (see Table 3, column *Clique*). It can be observed that an ILP approach addressing globally the objectives faces difficulties in attaining solutions.

Fortunately, given the order of priority dictated by our university on the criteria to be addressed, the objectives are not conflictual. We thus investigated a lexicographic scheme using ILP models. The lexicographic scheme achieves optimality at every stage for all instances and the obtained values are close to lower bounds. The final column in Table 9 gives the total computation time for the lexicographic scheme. As it can be seen, the optimality is attained in for instances within good computation times.

It has now been incorporated into the university's assignment process with a view to providing students with more acceptable individual timetables.

7. Conclusion

In this paper we have proposed preprocessing, lower bounds, and integer linear programming formulations, and we have investigated valid inequalities to address the real-world student scheduling problems that we face at Université de Technologie de Compiègne (UTC). The preprocessing that we use has proved to be effective in reducing instance sizes. We have introduced a new clique-based formulation that has proved to be more effective than other formulations in reducing the number of constraints. We have paid particular attention to criteria that help create friendlier individual timetables for students with reduced mobility. The lexicographic scheme, i.e., optimizing criteria in turn, achieves better results than minimizing a global objective function. Our successful approach is able to group sets of students into as few sections as possible for given learning activities, and to minimize moves that need to be made in a hurry by students moving between sites. This helps ensure that students have equitable individual timetables. A valid inequality that we propose has proved to be effective in helping to attain optimal solutions and in giving shorter computation times in cases where there are larger numbers of students that cannot be assigned. We obtained optimal solutions for all instances, and the numbers of non-assigned student are far fewer than those obtained with the heuristic used so far at UTC.

Solution values for the other criteria are equal or close to the lower bounds. Future research directions may focus on extending our proposed models to provide UTC administrators with a computer-aided tool for the timetable design stage, which would make estimates of the number of student that may potentially be assigned to a given set of activities.

Students and administrators alike have expressed their satisfaction with the timetables generated using our approach. Administrators can generate and use results directly, and the lexicographic solution approach has been incorporated into UTC's administrative procedures.

Acknowledgments

This work was carried out in the framework of the Labex MS2T, which was funded by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0 0 04-02). We would like to thank the *Direction Formation et Pédagogie* (DFP) and the *Direction des Systèmes d'Information* (DSI) at Université de Technologie de Compiègne for supporting this work. We would like to thank the reviewers for their insightful comments that helped improve the quality of this paper.

Abdullah, S. (2006). Heuristic approaches for university timetabling problems. PhD thesis, Doctoral dissertation, University of Nottingham.

Akkan, C. and Gülcü, A. (2018). A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem. Computers & Operations Research, 90:22–32.

Arbaoui, T., Boufflet, J.-P., and Moukrim, A. (2013). An analysis framework for examination timetabling. In Proceedings of the Sixth International Symposium on Combinatorial Search (SoCS 2013), pages 11–19. Leavenworth, WA, USA.

Arbaoui, T., Boufflet, J.-P., and Moukrim, A. (2015). Preprocessing and an improved mip model for examination timetabling. Annals of Operations Research, 229(1):19–40.

Arbaoui, T., Boufflet, J.-P., and Moukrim, A. (2019). Lower bounds and compact mathematical formulations for spacing soft constraints for university examination timetabling problems. Computers & Operations Research, 106:133–142.

Babaei, H., Karimpour, J., and Hadidi, A. (2015). A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86:43–59.

Burke, E. K. and Petrovic, S. (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140(2):266–280.

Cheng, E., Kruk, S., and Lipman, M. (2003). Flow formulations for the student scheduling problem. In Practice and Theory of Automated Timetabling IV, volume 2740, pages 299–309.

Dostert, M., Politz, A., and Schmitz, H. (2016). A complexity analysis and an algorithmic approach to student sectioning in existing timetables. Journal of Scheduling, 19(3):285–293.

Feldman, R. and Golumbic, M. C. (1989). Constraint satisfiability algorithms for interactive student scheduling. In Proceedings of the 11th international joint conference on Artificial intelligence, volume 2, pages 1010–1016.

Feldman, R. and Golumbic, M. C. (1990). Optimization algorithms for student scheduling via constraint satisfiability. The Computer Journal, 33(4):356–364.

Gilmore, P. C. and Hoffman, A. J. (2003). A characterization of comparability graphs and of interval graphs. In Selected Papers Of Alan J Hoffman: With Commentary, pages 65–74. World Scientific.

Golumbic, M. C. (2004). Algorithmic graph theory and perfect graphs, volume 57. Elsevier.

Hoshino, R. and Fabris, I. (2020). Optimizing student course preferences in school timetabling. In International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, pages 283–299. Springer.

Hossain, S. I., Akhand, M., Shuvo, M., Siddique, N., and Adeli, H. (2019). Optimization of university course scheduling problem using particle swarm optimization with selective search. Expert Systems with Applications, 127:9–24.

IBM (2020). Cplex User's Manual.

Kaur, M. and Saini, S. (2020). A review of metaheuristic techniques for solving university course timetabling problem. Advances in Information Communication Technology and Computing, pages 19–25.

Laporte, G. and Desroches, S. (1986). The problem of assigning students to course sections in a large engineering school. Computers & operations research, 13(4):387–394.

Lekkeikerker, C. and Boland, J. (1962). Representation of a finite graph by a set of intervals on the real line. Fundamenta Mathematicae, 51(1):45–64.

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. OR spectrum, 30(1):167–190.

McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., and Qu, R. (2007). The Second International Timetabling Competition: Examination Timetabling Track. Technical Report QUB/IEEE/TECH/ITC2007/Exam/v4.0, Queen's University, Belfast.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Di Gaspero, L., Qu, R., and Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing, 22(1):120–130.

Méndez-Díaz, I., Zabala, P., and Miranda-Bront, J. J. (2016). An ilp based heuristic for a generalization of the post-enrollment course timetabling problem. Computers & Operations Research, 76:195–207.

Miyaji, I., Ohno, K., and Mine, H. (1988). Solution method for partitioning students into groups. European Journal of Operational Research, 33(1):82–90.

Moon, J. W. and Moser, L. (1965). On cliques in graphs. Israel journal of Mathematics, 3(1):23–28.

Müller, T. and Murray, K. (2010). Comprehensive approach to student sectioning. Annals of Operations Research, 181(1):249–269.

Müller, T., Rudová, H., and Müllerová, Z. (2018). University course timetabling and International Timetabling Competition 2019. In Burke, E. K., Di Gaspero, L., McCollum, B., Musliu, N., and Özcan, E., editors, Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2018), pages 5–31.

Östergård, P. R. (2002). A fast algorithm for the maximum clique problem. Discrete Applied Mathematics, 120(1–3):197 – 207.

Pillay, N. (2014). A survey of school timetabling research. Annals of Operations Research, 218(1):261–293.

Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. Annals of Operations Research, 239(1):3–38.

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T., and Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. Journal of scheduling, 12(1):55–89.

Rose, D. J., Tarjan, R. E., and Lueker, G. S. (1976). Algorithmic aspects of vertex elimination on graphs. SIAM Journal on computing, 5(2):266–283.

Sabin, G. and Winter, G. (1986). The impact of automated timetabling on universities-a case study. Journal of the operational Research Society, 37(7):689–693.

Schaerf, A. (1999). A survey of automated timetabling. Artificial intelligence review, 13(2):87–127.

Shaohan, M. and Wallis, W. (1988). Maximal-clique partitions of interval graphs. Journal of the Australian Mathematical Society, 45(2):227–232.

Song, T., Liu, S., Tang, X., Peng, X., and Chen, M. (2018). An iterated local search algorithm for the university course timetabling problem. Applied Soft Computing, 68:597–608.

Thepphakorn, T. and Pongcharoen, P. (2020). Performance improvement strategies on cuckoo search algorithms for solving the university course timetabling problem. Expert Systems with Applications, 161:113732.

Tripathy, A. (1992). Computerised decision aid for timetabling—a case analysis. Discrete applied mathematics, 35(3):313–323.

Van den Broek, J., Hurkens, C., and Woeginger, G. (2009). Timetabling problems at the TU Eindhoven. European Journal of Operational Research, 196(3):877–885.

Vayssade, M. (1978). Une approche informatique pour résoudre des problèmes de partitionnement complexes. PhD thesis, Université de Technologie de Compiègne.

Woumans, G., De Boeck, L., Beliën, J., and Creemers, S. (2016). A column generation approach for solving the examination-timetabling problem. European Journal of Operational Research, 253:178–194.