Computers & Operations Research 76 (2016) 195–207

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

An ILP based heuristic for a generalization of the post-enrollment course timetabling problem

Isabel Méndez-Díaz [a], Paula Zabala [a,b], Juan José Miranda-Bront [a,b,*]

[a] Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina
[b] Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

ARTICLE INFO

Article history:
Received 5 August 2014
Received in revised form
30 November 2015
Accepted 25 June 2016
Available online 30 June 2016

Keywords:
Integer programming
Matheuristic
University timetabling

ABSTRACT

We consider a new timetabling problem arising from a real-world application in a private university in Buenos Aires, Argentina. In this paper we describe the problem in detail, which generalizes the Post-Enrollment Course Timetabling Problem (PECTP), propose an ILP model and a heuristic approach based on this formulation. This algorithm has been implemented and tested on instances obtained from real data, showing that the approach is feasible in practice and produces good quality solutions.

1. Introduction and literature review

Timetabling problems within the context of universities represent a very challenging task, where many different restrictions and demands must be satisfied by a feasible solution. During the last decade, this type of timetabling problems has received quite a lot of attention, partly due to the organization of three different competitions: the First International Timetabling Competition (ITC) in 2002, the Second ITC in 2007 and the Third ITC in 2011. As a result, a wide variety of methods and algorithms for different approaches to the problem has been proposed.

The ITC 2007 presented three tracks with different university-related timetabling problems: Examination Timetabling (ETP) [18], where the objective is to schedule exams along a time horizon while satisfying a set of constraints (see, e.g., [3,11]); Post Enrolment based Course Timetabling (PECTP) [18], where the objective is to schedule a set of events into rooms and time-slots (usually, a week) based on the selections made by the students; and Curriculum-based Course Timetabling (CCTP) [18], where the problem consists in scheduling a set of events to rooms and time-slots, but according to the curricula of the university (see, for example, [2,14,16]). Bettinelli et al. [1] provide a good overview on course timetabling problems, and Lübbecke [17] gives some further comments regarding practical and implementation issues.

The PECTP is defined by the following information, as stated in [18]: a set of events to be scheduled into a number of time-slots; a set of rooms with an associated capacity; a set of room-features that may be required by the events and satisfied by the rooms; a set of students who are enrolled in different combinations of events; a set of feasible time-slots for each of the events; and a set of precedence requirements among certain events. The objective is to assign the events to a room and a time-slot while satisfying the following *hard constraints*: every student must attend at most one event per time-slot; the room assigned to each event must have enough capacity and satisfy the features required by the corresponding event; at most one event is assigned to a room in any time-slot; events must be assigned to time-slots which are feasible; and where specified, events must be scheduled in the order established by the precedences. As regards the objective function, a set of *soft constraints* is defined, adding a penalization for each violation within the schedule: students should not be scheduled to attend an event in the last time-slot of a day; students should not attend three or more events in successive time-slots, and should not be required to attend only one event in a given day.

Several approaches have been proposed for the PECTP, mainly considering metaheuristics since in the ITC 2007 a strict time limit was imposed on the running time of the algorithms. Most of them are designed to tackle the problem in two or three stages, focusing first on the feasibility and then on the optimality of the generated timetable. Lewis [15] proposes a three-phase heuristic that uses Simulated Annealing (SA) for the last two in order to improve the generated schedule. Jat and Yang [12] propose a two phase approach using Genetic Algorithms and Tabu Search. Chiarandini et al. [9] propose a heuristic based on stochastic local search. Ceschia et al. [6] perform an extensive study for the PECTP by

* Corresponding author at: Departamento de Computación, Pabellón I, Ciudad Universitaria, C1428EGA, CABA, Argentina.
E-mail addresses: imendez@dc.uba.ar (I. Méndez-Díaz), pzabala@dc.uba.ar (P. Zabala), jmiranda@dc.uba.ar (J.J. Miranda-Bront).

http://dx.doi.org/10.1016/j.cor.2016.06.018
0305-0548/

considering several variations of the problem and propose a metaheuristic based on SA. Nothegger et al. [21] propose an ant colony optimization algorithm. Cambazard et al. [4] study a wide variety of approaches, including Constraint Programming (CP) and a list-coloring relaxation of the PECTP. Finally, van den Broek and Hurkens [23] propose an ILP based heuristic for the problem. They use a column-generation approach in a construction phase and then formulate an ILP model to refine the solution, focusing also on the soft constraints. They report competitive solutions compared to the five finalists in the competition. Due to the definition of the PECTP, many of these approaches apply a preprocessing phase in which conflicts among events are derived. For example, the set of events a particular student is enrolled in cannot be assigned to the same time-slot, since otherwise the schedule violates a hard constraint. In general, all approaches produced good computational results in the instances involved in the competition.

A different perspective of the problem is addressed in van den Broek et al. [24], where the authors study a real timetabling problem at TU Eindhoven. In this case, the weekly timetable is already given and students have a preference list of events and request to be assigned to a certain number of them. The objective is to assign students to events satisfying constraints that are similar to the PECTP (i.e., one event per time-slot, not exceeding capacity of the room, and minimum and maximum quota constraints) among some other constraints. They show that for some combinations of constraints the problem is $\mathcal{NP}$ – Complete and propose an Integer Linear Programming (ILP) formulation for the problem. Computational results show that the approach is effective for real instances considered.

Another problem similar to the PECTP is the so-called *student sectioning problem*, which consists in assigning students to particular sections (i.e., classes) based on their requests and satisfying several traditional constraints such as room and section capacity, and avoiding conflicts in students timetables due to overlapping assignments. Carter [5] describes the characteristics of the problem and provides the details of the scheduling system developed at the University of Waterloo. The timetable construction and the sectioning of students is divided in mainly three stages: a student preregistration for the next semester; the generation of an initial timetabling (involving both an automated and a manual stage); and the student scheduling, including a *drop*/*add* period. Students are willing to attend to a set of courses, which are composed by multiple sections. In this sense, the problem considers a more complex structure for courses than the PECTP, similarly to our problem. The problem also considers that a student is willing to take all the courses in the list provided. Murray et al. [20] and Muller and Murray [19] build upon this research, mainly using the method proposed by Carter [5] to construct the initial timetable. The former presents a framework for tackling the problem, including several practical considerations. Muller and Murray [19] tackle the overall problem, generating the initial timetable as in Carter [5] and providing further developments, including several new local search operators, for the student sectioning stage. In both cases, the research is motivated by its application in Purdue University.

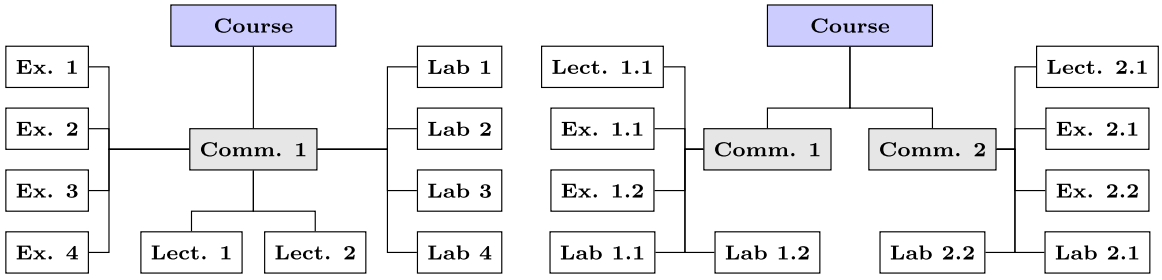
We further include a comparison with some other problems in the literature regarding timetables in educational institutions. The Balanced Academic Curriculum Problem (see, e.g., Chiarandini et al. [8]) aims to define at a general level the organization of courses for a university degree. The planning horizon is divided in years, where each of them is further divided into teaching terms where the courses can take place. Courses present precedences among them and load constraints are imposed on each teaching period, including both the number of courses assigned and the total credits involved in a teaching period. In a follow up paper, Ceschia et al. [7] consider a generalization of this problem.

Regarding the methodology, we briefly discuss a few approaches concerning the use of ILP techniques within more general frameworks. Sorensen and Dahms [22] propose a two stage decomposition heuristic based on an ILP formulation for a High School Timetabling Problem. Kristiansen et al. [13] consider an ILP for the High School Timetabling Problem, which is solved in two stages by means of a general purpose solver, but in this case resulting in an exact algorithm. The approach produces good results, obtaining 9 new best known solutions for the problem. Finally, Daskalaki and Birbas [10] consider a university timetabling problem where groups of students are enrolled in a set of courses, similarly to the PECTP. Standard operational constraints are considered, and a difference regarding the PECTP and with our problem is that all the requests of a student must be satisfied. The structure of the courses is, however, more general than in the PECTP and may include more than one type of class. The authors propose a two stage approach, where some of the heaviest operational constraints are relaxed in the first stage and then reconsidered in the second one, where smaller ILPs are formulated and solved for each day of the week independently.

Concerning the methodology proposed in Carter [5] for the student sectioning problem, firstly a *conflict matrix* is constructed, where each entry accounts for the number of students that have requested each pair of courses. In addition, due to the size of the problem considered, the students are first clustered based on the similarities among their courses' request and then a preliminary assignment to sections is performed, aiming always to minimize the expected number of conflicts. This step is referred as *homogeneous sectioning*. Using this information, the overall problem, consisting in approximately 3000 course sections and 17 000 students, is decomposed into several subproblems trying to group together sections with high interaction, which are then solved independently one at a time in decreasing order of difficulty. For each of these subproblems, different steps are considered sequentially. Firstly, an *automated course timetabling* step is considered where sections are assigned to time-slots aiming to minimize the total number of student conflicts and considering aggregated information regarding the room capacities by defining *room profiles*. This step is performed using a greedy heuristic and followed by a local search phase consisting of a 2-opt operator. Then, a *classroom assignment* step is performed using the information from the preregistration and including several constraints regarding type, distance and availability of the classrooms. After manual improvements made by the representatives of each university department, the original sectioning is discarded and the students are reassigned using the overall timetable generated in the previous steps. The students are sectioned one at a time in a two-pass fashion, considering in the first pass only some of the choices and in then the remaining ones. The objective is to prevent filling up courses only by students which registered earlier.

In this paper, we focus on a real-world application arising from a private university in Argentina which simultaneously involves timetabling events to time-slots and rooms as well as assigning students to a certain number of events chosen from their preference list. Similarly to the PECTP, the assignment of events to time-slots must satisfy certain constraints, which in turn depends on the assignment of students to events. In addition, courses have a particular hierarchical structure, similar to the student sectioning problem described before, that must be taken into account when performing the different assignments. This problem is named *Generalized Post-Enrollment Course Timetabling Problem* (GPECTP).

The contributions of this paper are threefold. Firstly, we study a problem with a direct and practical application that integrates, combines and tackles jointly two other problems from the related literature, namely the PECTP and the problem defined in van den



(a) Course with one commission (b) Course with two commissions

Fig. 1. Example of course structures. (a) Course with one commission. (b) Course with two commissions.

Broek et al. [24]. In addition to the hierarchical structure of the courses, as considered in the sectioning problem, in our problem only a certain number of the courses provided in the preference list of a student are to be assigned (indeed, we consider a lower and an upper bound on the number of assignments for each of them). Secondly, aiming to provide a solution methodology, we propose an ILP formulation that integrates the characteristics mentioned before. In order to use the algorithm within the context of the university, we develop a two-stage ILP based heuristic for GPECTP. Similarly to Carter [5], the first stage defines an assignment of classes to time-slots and then, based on these settings, the second stage assigns the students to the classes. However, there are some major differences regarding the methodology used in each of these stages. Firstly, Carter [5], once the problem is partitioned into smaller subproblems, generates only one timetable based on a pre-assignment of students to sections. In our case, we split the set of students into smaller subsets and by considering a similar idea to *room profiles* we generate several timetables per subset, obtaining as a result a multi-start approach for the problem. This is particularly important since, as we explain later in the paper, this helps in finding feasible and good quality timetables. Furthermore, the assignment of students to classes is done in an integrated fashion, considering as well the assignment of classes to rooms by means of an ILP formulation. This is an important characteristic since the assignments of students to classes, the assignment of classes to time-slots and the assignment of classes to classrooms are interdependent. Compared to the related literature, Carter [5] tackles these two problems sequentially: first the *classroom assignment* and then the *automated student sectioning*. A similar observation can be made regarding the approach in Muller and Murray [19], where the timetabling of classes and the assignment of classes to rooms is taken from the *initial sectioning* stage, although they propose several local search operators to find improved (feasible) solutions.

Finally, we report the results obtained on three real world instances, and analyze the benefits obtained by the university with the implementation of the algorithm. These instances are made public for further comparisons on the problem.

The paper is organized as follows. In Section 2 we present the context of the university and the description of GPECTP, including a discussion regarding its relation with known problems from the related literature. Then, Section 3 introduces the basic notation and definitions used throughout the paper and presents an ILP formulation for GPECTP. In Section 4 we present the ILP based heuristic developed to solve the problem. Computational results are shown in Section 5 and finally we draw some conclusions and future directions in Section 6.

2. Context and characteristics of the timetabling problem

The university offers 12 different programs of study, approximately 120 courses, and has 220 faculty members and 1200 registered students overall. The academic year is divided in two semesters that are scheduled independently. *Time-slots* refer to the available teaching periods within a week and remain unchanged during the whole semester.

In general, the curriculum of a program of study is not strict and often graduate students obtain their degrees by completing the curriculum in different ways. Courses are classified as mandatory or elective, depending on the program of study each student enrolls in. The former are courses that the student must attend (and, of course, pass) in order to obtain the corresponding degree. Elective courses are, on the other hand, grouped by topic; each student must then select from a restricted number of courses within each topic.

Each course consists of one or more modules corresponding to different types of *classes*. Such types are *lectures*, *exercise-classes* and *labs*. A course consists of exactly one of the following combinations of class types, hereafter named *patterns*: lectures, or lectures and exercise-classes, or lectures, exercise-classes and labs. It should be mentioned that each of these classes consists of one or more *events*, which have to be scheduled within the week and may have certain restrictions regarding its periodicity and time availability. For example, the university could require some courses to be scheduled within certain hours of the day due to academic and more general policies.

In addition, a course is composed of one or several *commissions*, which are instances of the same course. Each of these commissions has its own set of classes and all of them follow exactly the same pattern as regards the modules of the course. A commission may have more than one instance of each type of class. A student is required to be enrolled in only one commission of a course and all classes enrolled in must belong to the same commission.

Fig. 1 illustrates this. In Fig. 1(a) the course consists of only one commission and the pattern is defined by lectures, exercise-classes and labs. Then, the pattern for a student enrolled in the course can be any combination of one lecture, one exercise class and one lab included in Comm. 1. Fig. 1(b) shows a course with two commissions. A student has to be enrolled either in Comm. 1 or Comm. 2, and within each commission any combination of classes following the pattern is feasible. For example, patterns (Lect. 1.1, Ex. 1.2, Lab. 1.1) and (Lect. 1.1, Ex. 1.1, Lab 1.1) are feasible for a student enrolled in Commission 1, but pattern (Lect. 1.1, Ex. 1.1, Lab. 2.2) is infeasible.

By the end of each semester, the academic offer for the next semester is decided and each department assigns the faculty to the courses. Together with this information, every professor informs the administrative staff about the time availability and a maximum number of days per week in which the assigned classes have to be scheduled. This information, combined with the requirements of the course, results in the final available time-slots for each class.

As regards infrastructure, the university has 30 rooms with seating capacity ranging from 8 to 110 people. Classification of rooms is based on their attributes and special features like media equipment, computers, etc. Regarding the classes, some of them

can be given in any room while others require some specific attributes (for example, availability of computers).

Until the system was implemented, the construction of the timetable was very time-consuming and iterative. Once all the information was gathered, the administrative office was responsible for the timetabling process. Based on the knowledge about potential students for each course, the scheduler manually attempted to timetable the courses avoiding conflicts and assuring that every student had the possibility to attend a required number of courses. It should be mentioned that the latter aspect is particularly important because, being a private university, every student having a reasonably good academic performance must be able to obtain the degree within the expected time and to avoid paying extra tuition fees.

From an initial assignment, if there were conflicts or some student could not attend the required courses, the scheduler asked some of the faculty staff members to accept a different time-slot and repeated the laborious process in order to obtain an improved scheduling. Once the timetabling was complete, students were informed about the courses enrolled in. As a result of this process, at the beginning of the semester, long lines of students were formed at the administration office requesting possible changes in the courses they have been assigned to. These requests were again delivered to the scheduler who tried, if possible, to satisfy their requests by making some shiftings.

The process had three major drawbacks. Firstly, it was completely inefficient and time-consuming since it is not humanely possible to consider all the alternatives. Secondly, the whole process and the responsibility of starting the semester with the timetable on time depended mainly on the scheduler. Regardless of his/her outstanding performance, it entailed an excessive level of pressure and represented a weak point from the organizational perspective, since any unexpected event (for example, the scheduler getting sick) may jeopardize the beginning of the semester. Finally, the students were required to have full-time status and their assignments to courses were compulsive, they did not have the possibility of choosing which courses to be enrolled in for the semester.

Due to this complex situation, the university decided to develop a system to assist the scheduler with the timetabling process. Furthermore, aiming to overcome the third drawback mentioned before, the whole timetabling process should follow a demand-driven approach. Given the context and the university requirements, our suggestion is to provide students with a set of courses they are willing to take, a *preference list*, that includes more possibilities than the maximum number of courses they are allowed to take, and to let them rank the courses according to their preferences. We also establish for each student a minimum number of courses that must be selected from its preference list, in order to guarantee a minimum degree of possible academic progress for the semester. In addition, the university ranks the students according to their academic records and other criteria, which are also taken into account in the timetabling process. Then, the optimization problem we are tackling involves maximizing the global weighted preference (a combination of students preferences and rank performances) while assigning lessons to the time-slots, taking into account the choices made by the students and several other university resource constraints.

A difference regarding the GPECTP and other timetabling problems is that we are requested to consider all operational and resource constraints as hard. This of course has an impact on the difficulty of finding a feasible timetable satisfying all constraints. The university authorities give priority to all constraints and decide that, in case no feasible solution is found, the system should be used on less restrictive scenarios (for example, by expanding the available time-slots for the academic staff or reducing the minimum number of courses to be assigned for each student).

As mentioned in the introduction, the GPECTP considers simultaneous characteristics from the PECTP and from the research in van den Broek et al. [24]. The latter takes into account preference lists, but assumes that the assignment of events to time-slots has already been done and that it is an input to the problem. On the other hand, if the set of courses provided by a student represents indeed the courses in which she is enrolled, then the problem becomes the PECTP – despite some differences in the particular constraints considered. However, in our case, the assignments of courses to students is one of the outcomes of the problem. More precisely, in the PECTP the events to which a student is to be assigned are known a priori. Therefore, in the list-coloring approach proposed by Cambazard et al. [4] the set of events chosen by a student define, as they name it, a *student clique*. All those events have to be scheduled in different time-slots, enabling a preprocessing phase where conflicts among events can be derived and used within the optimization.

In the case of the GPECTP this analysis is not possible due to the following two reasons. Firstly, since the preference list of a student usually contains more courses (and, therefore, events) than the maximum number of courses allowed to take simultaneously, then the conflicts among the events depend on the courses effectively enrolled in, which in turn depends on the time-slots assigned to the events. In addition, even assuming that we know exactly the courses to be assigned to a particular student, the hierarchical structure of the courses and its composition in terms of commissions have a similar effect, since the events effectively enrolled in are not known a priori and, furthermore, it is one of the outputs of the GPECTP.

In the next section we present the notation and the formal definitions used throughout the paper as well as the ILP formulation for the problem. Since the problem involves a large number of constraints, they are explained in detail while formulating the model.

3. ILP formulation

3.1. Definitions and notation

We begin by introducing some basic definitions and the notation used to describe the problem. For clarity reasons, we group the definitions according to the entity (students, courses, and time-slots). Given the large number of parameters and definitions, for notation purposes we adopted the following criteria. Definitions are referred with superscripts, while decision variables with subscripts. Sets are identified by capital letters, and weights used in the objective function by Greek letters.

3.1.1. Days and time-slots

As mentioned before, the timetable considers a week. The week is modeled as a collection of consecutive days; each of them is defined by a collection of consecutive time-slots where events are scheduled. In practice, the week consists of six days, Monday to Saturday. Saturday is defined by two time-slots, while the remaining working days are defined by five time-slots. Time-slots are numbered in ascending order following their position within the week.

$\mathcal{D} = \{1, \ldots, \bar{d}\}$	Set of working days
S^d	Set of time-lots defining working day $d \in \mathcal{D}$
$\underline{s}^d$, $\bar{s}^d$	First and last time-slot of working day $d \in \mathcal{D}$, respectively
$\mathcal{S} = \cup_{d \in \mathcal{D}} S^d$	Set of time-slots

3.1.2. Rooms

For each room, we identify two attributes: its seating capacity and its type, which is an internal classification made by the university.

$\mathcal{R}$	set of rooms
$Types$	set of room's type
cap^r	capacity of room $r \in \mathcal{R}$
nr^i	number of rooms of type i for $i \in Types$
$Classes^i$	classes that could be assigned to rooms of type i
t^r	type of room $r \in \mathcal{R}$

3.1.3. Courses

The definitions introduced below represent the hierarchical structure of courses introduced before. Each event included in a class is assumed to fit exactly in one time-slot and all classes have a periodicity, meaning that a class must be given in a determined number of days, with a fixed number of events per day. Let c be a class, with V^c its set of events, nd^c its weekly periodicity and nv^c the number of events per day. Then, $|V^c| = nd^c \times nv^c$. Finally, it should be mentioned that the order in which the events of a class are scheduled is not relevant and, therefore, fixed *a priori*.

$\mathcal{M}$	set of courses
$\mathcal{C}$	set of classes
$\mathcal{G}$	set of courses' group
Q^m	commissions of course $m \in \mathcal{M}$
T^m	lecture classes of course $m \in \mathcal{M}$
Cl^m	set of classes of course $m \in \mathcal{M}$
$Lec^q \subseteq \mathcal{C}$	lecture classes of commission q
$Ex^q \subseteq \mathcal{C}$	exercise classes of commission q
$Lab^q \subseteq \mathcal{C}$	lab classes of commission q
$Slots^c \subseteq \mathcal{S}$	feasible time-slots for class $c \in \mathcal{C}$
V^c	set of events of class $c \in \mathcal{C}$
nd^c	weekly periodicity of class $c \in \mathcal{C}$
nv^c	number of consecutive events on a same day of class $c \in \mathcal{C}$
$Init^c$	initial events for each of the nd^c blocks of consecutive events for class $c \in \mathcal{C}$
$R^c \subseteq \mathcal{R}$	set of admissible rooms for class $c \in \mathcal{C}$
q^c	quota of class $c \in \mathcal{C}$
$nreq^m$	number of students that must attend course $m \in \mathcal{M}$

3.1.4. Students

Each student is asked to provide a preference list of courses, from which a subset with a minimum and a maximum number of courses will be selected. Course assignment for the semester will be based on this subset. Within the preference list, some courses can be marked as *required*, meaning that the student must be assigned to at least one commission of that course. This is done by using a particular preference value.

The remaining courses are classified as either mandatory or elective. Elective courses belong to exactly one group, for which there is also a minimum and a maximum number of courses that can be assigned to a student. Although groups are defined by program of study, we adopt a particular definition for each student in order to capture the nature of students' different situations. For example, a student may have already passed some courses of a particular group, but still be able to select some others. In this sense, given a student $a \in \mathcal{A}$, G^a is the set containing the definitions of each group. We assume that two groups defined for a student cannot share courses.

Finally, we note that the higher the ranking for a student, the higher the priority.

$\mathcal{A}$	student set
rk^a	academic performance weight for student $a \in \mathcal{A}$
$\bar{m}^a$	maximum number of courses to be assigned to student $a \in \mathcal{A}$
$\underline{m}^a$	minimum number of courses to be assigned to student $a \in \mathcal{A}$
$M^a \subseteq \mathcal{M}$	set of courses chosen by the student $a \in \mathcal{A}$
G^a	set of groups definitions chosen by the student $a \in \mathcal{A}$
w^{am}	preference value for course $m \in \mathcal{M}$ given by student $a \in \mathcal{A}$
$\bar{m}^{ag}$	maximum number of courses to be assigned to student $a \in \mathcal{A}$ from group $g \in G^a$
Sl^a	available slots for student $a \in \mathcal{A}$

3.1.5. Faculty members

Finally, we introduce some notation to refer to faculty members. Together with the available time-slots, each member indicates also a maximum number of days he/she is willing to use for teaching purposes. Then, all classes assigned to this member should be scheduled taking this into account.

$\mathcal{F}$	set of faculty members
$S^f \subseteq \mathcal{S}$	available time-slots of $f \in \mathcal{F}$
$C^f \subseteq \mathcal{C}$	classes taught by $f \in \mathcal{F}$
$\bar{ndays}^f$	maximum number of days per week $f \in \mathcal{F}$ wants to give lessons

Although the assignment of the teaching staff to the classes is assumed to be done at an earlier stage, it affects the time-slots where a given class can be assigned. Therefore, the set of feasible time-slots of a class $c \in \mathcal{C}$ must account also for the available time-slots of the staff assigned, i.e., $\cap_{f \in \mathcal{F}, c \in C^f} S^f \cap Slots^c$. For notation purposes, we will refer to this set as $Slots^c$.

3.2. Decision variables

We consider six sets of binary variables to model the different constraints of the timetable. Binary variable x_{ac} takes value 1 iff student $a \in \mathcal{A}$ is assigned to class $c \in Cl^m$, for $m \in M^a$. We further define binary variable xs_{acs} that takes value 1 iff class c assigned to student a is scheduled to time-slot $s \in Slots^c \cap Sl^a$. These two sets of variables let us handle the student-class assignment aspect of the timetable.

Given a class $c \in \mathcal{C}$ and an event $v \in V^c$, binary variable y_{cvs} takes value 1 iff v is assigned to time-slot $s \in Slots^c$. In addition, binary variable yr_{crs} takes value 1 iff class $c \in \mathcal{C}$ is assigned to room $r \in R^c$ in time-slot $s \in Slots^c$ and binary variable w_c takes value 1 iff class $c \in \mathcal{C}$ is scheduled in the timetable. These three sets of variables allow us to model the assignment of classes to rooms and time-slots.

Finally, we define variables to model restrictions imposed by the faculty members. Binary variables z_{fd} take value 1 iff professor $f \in \mathcal{F}$ has a class assigned to day $d \in \mathcal{D}$.

We next describe in detail the constraints that define a feasible timetable.

3.3. Modeling problem constraints

The formulation we present next models the usual constraints that appear in academic timetabling problems together with some particular constraints considered in this case. Specific details are given with each type of constraints.

3.3.1. Student-course assignment constraints

Each student informs a preference list of courses he/she is willing to take during the semester, from which a subset containing at least $\underline{m}^a$ and at most $\bar{m}^a$ has to be assigned. Groups are managed in a similar fashion. For each assigned course, the pattern must be feasible and the quota for each class must not be exceeded.

Regarding the structure of the courses and its correct assignment to a commission, we note that feasible patterns always include a lecture class. Thus, we use this type of classes to manage the assignment of a student to a commission and then establish the necessary relations to conform a feasible pattern. For simplicity reasons, a commission has an empty set of classes of a particular class-type if and only if the pattern for the course does not include them. Finally, classes have an assigned quota, which must not be exceeded:

$$\sum_{c\in T^m} x_{ac} \le 1 \quad a\in\mathcal{A},\ m\in M^a \tag{1}$$

$$\sum_{c\in Lec^q} x_{ac} = \sum_{c\in Ex^q} x_{ac} \quad a\in\mathcal{A},\ m\in M^a,\ q\in Q^m,\ Ex^q\neq\varnothing \tag{2}$$

$$\sum_{c\in Lec^q} x_{ac} = \sum_{c\in Lab^q} x_{ac} \quad a\in\mathcal{A},\ m\in M^a,\ q\in Q^m,\ Lab^q\neq\varnothing \tag{3}$$

$$\underline{m}^a \le \sum_{\substack{c\in T^m\\ m\in M^a}} x_{ac} \le \bar{m}^a \quad a\in\mathcal{A} \tag{4}$$

$$\sum_{\substack{c\in T^m\\ m\in g}} x_{ac} \le \bar{m}^{ag} \quad a\in\mathcal{A},\ g\in G^a \tag{5}$$

Constraints (1) establish that each student must be assigned to at most one lecture class of the courses in the preference list. If a course is marked as required for the student, the constraint is imposed by equality. Constraints (2) and (3) force the assignment to a course to be a feasible pattern. Constraints (4) restrict the minimum and maximum number of courses that can be assigned to a student, and constraints (5) establish an analogous restriction within groups defined by the student.

3.3.2. Class-slot feasibility constraints

The assignment of classes to time-slots has to satisfy different types of restrictions. For a schedule to be feasible, the block structure of the class has to be satisfied and each event has to be assigned to exactly one time-slot, without overlaps among them. In addition, some of the classes consisting of two or more blocks cannot be assigned to consecutive days (due to some pedagogical preferences):

$$\sum_{a\in\mathcal{A}} x_{ac} \le q^c w_c \quad c\in C \tag{6}$$

$$\sum_{s\in Slots^c} y_{cvs} = w_c \quad c\in C,\ v\in V^c \tag{7}$$

$$\sum_{s\in S^d}\sum_{v\in Init^c} y_{cvs} \le 1 \quad d\in\mathcal{D},\ c\in C \tag{8}$$

$$\sum_{s=s_0+1}^{s_0+nv^c-1}\sum_{v=v_i+1}^{v_i+nv^c-1} y_{cvs} = (nv^c-1)y_{cv_is_0} \quad d\in\mathcal{D},\ s_0\in Slots^c,\ s_0\le\bar{s}^d-nv^c,\ c\in C,\ v_i\in Init^c \tag{9}$$

$$\sum_{s\in S^d}\sum_{v\in Init^c} y_{cvs} + \sum_{s\in S^{\bar{d}+1}}\sum_{v\in Init^c} y_{cvs} \le 1 \quad d\in\mathcal{D},\ d\neq\bar{d},\ c\in C \tag{10}$$

Constraints (6) establish that if a class is scheduled, then its quota must not be exceeded. Constraints (7) schedule the events of a class to exactly one feasible time-slot if the class is scheduled, and constraints (8) establish that at most one event of each block can be scheduled within one day. In addition, constraints (9) ensure that consecutive events in a class block are scheduled to consecutive times-slots within the same day. Note that these last three constraints model the correct assignment of events to time-slots in terms of slots overlapping conflicts. Finally, inequalities (10) establish that different blocks of a class must be scheduled in non-consecutive days. Note that it is sufficient to express these constraints only in terms of the initial event of each block, since constraints (9) impose the assignment of the entire block to consecutive time-slots within the same day. It is worth to note that constraints (10) apply just for classes with non-consecutive days requirement. For these classes, constraints (8) can be omitted since they are dominated by (10).

3.3.3. Class-room feasibility constraints

The assignment of classes to rooms is feasible if it satisfies some standard restrictions, mainly avoiding exceeding the capacity of the room and overlappings in a time-slot. Constraints (11) establish that if a class is assigned then all of its events must be assigned to exactly one room in the corresponding time-slot. Constraints (12) force that at most one event is assigned to a room in a time-slot and constraints (13) assures that the capacity of a room is not exceeded. In addition to these feasibility constraints, inequalities (14) force different events in a block to be assigned to the same room. This is similar to the *room stability* constraint in related university timetabling problems:

$$\sum_{r\in R^c} yr_{crs} = \sum_{v\in V^c} y_{cvs} \quad c\in C,\ s\in Slots^c \tag{11}$$

$$\sum_{c\in C, r\in R^c} yr_{crs} \le 1 \quad r\in\mathcal{R},\ s\in\mathcal{S} \tag{12}$$

$$\sum_{a\in\mathcal{A}} xs_{acs} \le \sum_{r\in R^c} cap^r yr_{crs} \quad c\in C,\ s\in Slots^c \tag{13}$$

$$(nv^c-1)(y_{cv_is_0}+yr_{crs_0}-1) \le \sum_{s=s_0+1}^{s_0+nv^c-1} yr_{crs} \quad d\in\mathcal{D},\ s_0\in Slots^c,\ s_0\le\bar{s}^d-nv^c,\ c\in C,\ v_i\in Init^c,\ r\in R^c \tag{14}$$

3.3.4. Student-time-slot feasibility constraints

These constraints are necessary to guarantee that the timetable is feasible for the students. Similarly to the PECTP we need to prevent a student from being assigned to more than one class in a particular time-slot. In addition, the timetable specifically requires the conditions are satisfied regarding the number of classes in the course assigned to and the distribution of the classes assigned to a student on a given day:

$$y_{cvs} + x_{ac} - xs_{acs} \le 1 \quad a\in\mathcal{A},\ m\in M^a,\ c\in Cl^m,\ v\in V^c,\ s\in Slots^c \tag{15}$$

$$2xs_{acs} - \sum_{v\in V^c} y_{cvs} - x_{ac} \le 0 \quad a\in\mathcal{A},\ m\in M^a,\ c\in Cl^m,\ s\in Slots^c \tag{16}$$

$$\sum_{\substack{c\in Cl^m\\ m\in M^a}} xs_{acs} \le 1 \quad a\in\mathcal{A},\ s\in\mathcal{S} \tag{17}$$

$$\sum_{c\in Cl^m}\sum_{s\in S^d} xs_{acs} \le 2 \quad a\in\mathcal{A},\ m\in M^a,\ d\in\mathcal{D} \tag{18}$$

$$\sum_{m\in M^a}\sum_{c\in Cl^m} xs_{ac\underline{s}^d} + \sum_{m\in M^a}\sum_{c\in Cl^m} xs_{ac\bar{s}^d} - \sum_{\substack{s\in S^d \\ s\neq \underline{s}^d, s^d}}\sum_{m\in M^a}\sum_{c\in Cl^m} xs_{acs} \leq 1 \quad a\in\mathcal{A},\ d\in\mathcal{D} \tag{19}$$

Inequalities (15) and (16) state the correspondence between variables x_{ac}, y_{cvs} and variables xs_{acs}. Constraints (17) force that students are assigned to not more than one event in a time-slot, in order to provide each student with a feasible schedule, and constraints (18) establish that within a day a student cannot have scheduled more than two events belonging to the same course. This characteristic is imposed by the university due to pedagogical aspects.

Finally, although students are assumed to have full-time status, the university decided not to generate assignments for students in which they are required to attend a class in the first and last time-slot of a day, without having activities in between. For this purpose, constraints (19) establish that if student events are scheduled in the first and last time-slot of a single day, then at least one of the intermediate time-slots must have an event assigned as well.

3.3.5. Class-faculty feasibility constraints

The model must consider the feasibility of the schedule regarding the information provided by the faculty members, based on their availability and the overlap of events assigned to the same professor. For this purpose, constraints (20) prevent the model from considering solutions with more than one event assigned to the same professor in the same time-slot. In addition, inequalities (21) and (22) are necessary to satisfy the restrictions imposed by each professor regarding the maximum number of days they are willing to give classes. In particular, inequalities (21) activate variables z_{fd} if professor f has at least one block of events assigned to day d, and inequalities (22) limit the number of these variables that can take value 1 based on the maximum number informed to the administration office:

$$\sum_{c\in C^f}\sum_{v\in V^c} y_{cvs} \leq 1 \quad f\in\mathcal{F},\ s\in S^f \tag{20}$$

$$\sum_{c\in C^f}\sum_{v\in Init^c}\sum_{s\in S^d} y_{cvs} \leq |S^d| z_{fd} \quad f\in\mathcal{F},\ d\in\mathcal{D} \tag{21}$$

$$\sum_{d\in\mathcal{D}} z_{fd} \leq n\bar{d}ays^f \quad f\in\mathcal{F} \tag{22}$$

3.4. Objective function and ILP formulation

In order to complete the description of the ILP formulation, we model the objective function. For each student $a\in\mathcal{A}$, we consider a weighted sum between the preference value of a course $m\in M^a$ and his/her rank priority. In particular, given the structure of the courses and the feasible patterns defined, a student $a\in\mathcal{A}$ is enrolled in a course if and only he/she is enrolled in one of its lectures. Taking into account the decision variables x_{ac}, we further define $wc^{ac} = w^{am}$ for $a\in\mathcal{A}$, $m\in M^a$, $c\in T^m$. Thus, considering nonnegative weights α and β, the ILP formulation maximizes the total weighted preference for the assignments of students to courses subject to the feasibility constraints:

$$(\text{GPECTP}-\text{ILP}) \quad \max z_{ILP} = \sum_{a\in\mathcal{A}}\sum_{\substack{c\in T^m \\ m\in M^a}} (\alpha rk^a + \beta wc^{ac}) x_{ac}$$

$$\text{s. t.:} \quad (1)\text{–}(22)$$

$$x_{ac}, xs_{acs}, y_{cvs}, yr_{crs}, z_{fd}, w_c \in \{0,1\} \tag{23}$$

This formulation has a large number of both variables and constraints for real-world instances, making it difficult to be solved by a general purpose ILP solver such as CPLEX. Indeed, on preliminary computational experiments, CPLEX default algorithm showed difficulties firstly in finding feasible solutions and, also, in improving the upper bound, even after enumerating a large number of nodes. Finding a feasible solution is a difficult problem due to the large number of operational constraints imposed. Regarding the upper bounds, since both the schedule of events into time-slots and the assignment of students to courses are tackled together, traditional techniques used in ILP algorithms are not as powerful as in other problems. For example, the preprocessing phase applied to the PECTP cannot be directly applied in the GPECTP, since incompatibility of events depends on the assignments of students to courses, which in turn depends on the assignments of events to time-slots. A similar observation can be made regarding the standard branching rules.

4. Solution methodology

4.1. Overall approach

Due to the reasons mentioned before, we develop a two stage heuristic procedure based on the formulation described in the previous section. The sketch of the procedure is as follows. In the first stage we generate assignments of classes to time-slots considering a subset of students and a relaxation of the ILP formulation, named GPECTP-RED. Then, for each generated assignment, we proceed to solve the model GPECTP-ILP but fixing some of the variables to the values of the solution obtained in the first stage. Finally, the best timetable is selected as the final solution.

As mentioned before, the idea behind the second stage is rather simple: given an output produced by GPECTP-RED, fix the assignment of classes to time-slots and solve the formulation GPECTP-ILP. However, considering only one assignment of classes to time-slots from the first stage may result in a poor quality solution or, even worse, not being able to find a feasible timetable satisfying all the constraints. Thus, we randomly generate l reduced subsets of students $\bar{A}_1,\ldots,\bar{A}_l$ to be used in the first stage. By considering these subsets we are introducing in the first stage some of the information used in the second stage. Indeed, despite some relaxed constraints and the objective function which was partially reformulated, solving GPECTP-RED is somehow similar to solving GPECTP-ILP but on a smaller scale.

In addition, for each of these subsets we decided to consider more than one solution in the first stage, applying the second stage for each of them. This is done by means of CPLEX routine *populate*, which given an ILP formulation tries to generate alternative optimal or sub-optimal solutions and to store them in a pool. Under this scheme, the first stage acts as a sort of construction phase and the second stage as a local search procedure.

For the construction of the subsets $\bar{A}_1,\ldots,\bar{A}_l$ we adopt the following strategy. Initially, all subsets contain the following students:

1. all students that are in the K% top ranking, and
2. for each student having at least one required subject not included in 1, we include an artificial student considering only the required subjects.

Then, students not considered in the previous definition are randomly assigned to exactly one of the l subsets. It should be mentioned that the artificial student included in 2 is replaced if the original student is assigned to the subset.

Before describing the heuristic in detail, we first discuss a particular issue identified during the deployment of the algorithm.

Although the university required the construction of the timetable based on the ranking of the students, in practice they analyzed the quality of the timetables also considering the percentage of assignments of students to courses. Formally, let $\pi^* = (x^*, xs^*, y^*, z^*, w^*, rc^*, rc^*)$ be the characteristic vector of a feasible solution of GPECTP-ILP, then they are also interested in the following metrics:

$$r = \frac{\sum_{a\in\mathcal{A}} \sum_{m\in M^a} \sum_{c\in T^m} x^*_{ac}}{\sum_{a\in\mathcal{A}} \bar{m}^a},$$

which gives the ratio between the number of assignments of students to courses in the solution and the possible maximum number of assignments, where the timetable assigns to each student the maximum possible number of courses. Considering the objective function defined in GPECTP-ILP, solutions with a better objective value do not necessarily have a higher value of r. Therefore, we adopted the following strategy: whenever a new current solution is found during the second stage, it is stored and reported once the execution is completed. With this information, it is possible to analyze more than one timetable and to select the one the scheduler prefers.

In Algorithm 1 we show the sketch of the overall procedure. As input parameters, in addition to the definition of the instance, the user chooses the number of subsets l and the number of solutions (N) to be generated for each subset in the first stage, the time limit for the second stage (*TILIM*), and the threshold (K) to compute the restricted subsets $\bar{A}$. The latter is a value between 0 and 100, representing the K percent of the students with the highest ranking.

Algorithm 1. Overall algorithm.

Input: Instance, l, N, *TILIM*, K
Output: Set Δ of feasible timetables for GPECTP.
1. Formulate GPECTP-ILP model. Compute UB as the optimal value of its LP relaxation to report a quality metric on the final solutions.
2. Compute $\bar{A}_1, \ldots, \bar{A}_l$ according to K.
3. Set $BESTLB = -\infty$ and $\Delta = \emptyset$ as the timetable solution pool.
4. for $j = 1, \ldots, l$ **do:**
5. Formulate the phase-one ILP for subset $\bar{A}_j$, solve it and generate N feasible assignments of classes to time-slots using the *populate* routine. Let Γ_j be the solution pool.
6. for $i \in \Gamma_j$ **do:**
7. Let $x^{(i)}$ be the i-th solution from the first phase. Construct the ILP model and fix variables assigning classes to time-slots. Set $UB^{(i)}$ as the objective function of the LP relaxation.
8. If $UB^{(i)} \le BESTLB$, then skip this solution and return to 7 with solution $i+1$ of Γ.
9. Otherwise, solve current ILP for (at most) *TILIM* seconds. Let $z^{(i)}$ be the objective function of the best solution found during the optimization of the second stage. If $z^{(i)} > BESTLB$, then set $BESTLB = z^{(i)}$, store the solution in Δ and continue with solution $i+1$ of Γ_j.
10. end for
11. end for
12. Report all solutions in Δ.

4.2. Stage 1: Class-time-slot construction

To generate a feasible assignment of classes to time-slots we simplify the problem in two different ways. Firstly, we consider a restricted set of students $\bar{A} \subseteq \mathcal{A}$ in order to reduce the size of the model. Secondly, we discard the room capacity constraints of the model and include only one quantitative restriction. Let nr^i be the number of rooms of type $i \in Types$ and $Classes^i \subseteq C$ the set of classes that can be assigned to rooms of type i, then for each time-slot the number of assigned classes of type i cannot exceed nr^i. Clearly, this is a relaxation of the overall problem and does not guarantee the fixed seating capacity of the rooms.

Since in this stage we are only interested in the assignment of classes to time-slots, considering a restricted set of students $\bar{A}$ may result in timetables with an undesired characteristic: for courses that are not required by any student in $\bar{A}$, the events of these courses may overlap in such a way that no feasible assignment can be done. This may lead to myopic assignments of classes to time-slots, which are only reasonable (or even feasible) in the case of students included in $\bar{A}$, resulting in poor quality overall solutions in the second stage. On the other hand, forcing the schedule to assign each class of the course to non-overlapping time-slots may be too demanding in terms of feasibility, specially for courses having several commissions.

To overcome this issue, we impose on the ILP formulation that from this stage at least one feasible pattern must be assigned to non-overlapping time-slots. For courses where the pattern is just a lecture class, clearly the feasibility is imposed by the inequalities set by the original formulation. For courses where the pattern is lecture and exercise-class, we impose that each lecture must have at least one feasible exercise-class to conform a pattern, and vice versa. A similar idea is enforced for classes where feasible patterns must have one of each type of class, i.e., lecture, exercise-class and lab. In the latter, we consider adding these constraints in pairs (for example, one lecture with one lab) or in groups of three (for a lecture there must exist an exercise-class and a lab that together consist in a feasible pattern).

We begin by modeling the first case, where the feasible patterns are lecture and exercise-class. Formally, let $m \in \mathcal{M}$, $q \in Q^m$ such that $Lab^q = \emptyset$, and consider $c_1 \in Lec^q$, $c_2 \in Ex^q$, we define binary variables $rc_{c_1c_2}$ that take value one iff class c_1 is *related* to class c_2, where the term related means that they form a feasible pattern. In addition, we consider variables y_{cvs}, w_c, x_{ac} and xs_{acs} defined previously and restricted, when necessary, to the set $\bar{A}$. For all these inequalities, unless otherwise stated, we are assuming $Lec^q, Ex^q \neq \emptyset$ and $Lab^q = \emptyset$:

$$\sum_{c\in Ex^q} rc_{c_1c} \ge 1 \quad c_1 \in Lec^q,\ q \in Q^m,\ m \in \mathcal{M} \tag{24}$$

$$\sum_{c\in Lec^q} rc_{cc_2} \ge 1 \quad c_2 \in Ex^q,\ q \in Q^m,\ m \in \mathcal{M} \tag{25}$$

$$rc_{cc'} + w_c \le 1 + w_{c'} \quad c \in Lec^q,\ c' \in Ex^q,\ q \in Q^m,\ m \in \mathcal{M} \tag{26}$$

$$rc_{cc'} + w_{c'} \le 1 + w_c \quad c \in Lec^q,\ c' \in Ex^q,\ q \in Q^m,\ m \in \mathcal{M} \tag{27}$$

$$\sum_{v\in V^c_1} y_{c_1vs} + \sum_{v\in V^c_2} y_{c_2vs} \le 2 - rc_{c_1c_2} \quad c_1 \in Lec^q,\ c_2 \in Ex^q,\ q \in Q^m,\ m \in \mathcal{M} \tag{28}$$

$$|Ex^q| \sum_{c\in Lec^q} w_c \ge \sum_{c\in Ex^q} w_c \quad m \in \mathcal{M},\ q \in Q^m \tag{29}$$

$$\sum_{c\in Classes^i} \sum_{v\in V^c} y_{cvs} \le nr^i \quad i \in Types,\ s \in \mathcal{S} \tag{30}$$

Constraints (24) and (25) establish that every lecture must be related to at least one exercise-class, and vice versa. Based on this relation, constraints (26) and (27) enforce that if one class is scheduled, then its related classes are scheduled too. Constraints (28) forbid related classes to be assigned to the same time-slot.

Constraints (29) establish that if no lecture class is scheduled for a particular course, then neither are exercise-classes since there is no possibility of constructing a feasible pattern. Constraints (30) establish that the number of classes requiring a particular type of room does not exceed the availability of such room type.

We consider a similar approach for courses having as a feasible pattern one class of each type, i.e., one lecture, one exercise-class and one lab belonging to the same commission. Let rc_{tpl} be a binary variable that takes value one iff classes $t \in Lec^q$, $p \in Ex^q$ and $l \in Lab^q$ are related, for $q \in Q^m$, $m \in \mathcal{M}$. The combination of classes t, p and l represents a feasible pattern for a student enrolled in commission q of course m. The following inequalities guarantee that the combination is feasible. Similarly to the previous case, we assume that Lec^q, Ex^q and Lab^q are non-empty sets:

$$2rc_{tpl} + 2w_t - w_p - w_l \le 2 \quad t \in Lec^q, p \in Ex^q, l \in Lab^q, q \in Q^m, m \in \mathcal{M} \tag{31}$$

$$\sum_{p \in Ex^q, l \in Lab^q} rc_{tpl} \ge 1 \quad t \in Lec^q, q \in Q^m, m \in \mathcal{M} \tag{32}$$

$$\sum_{t \in Lec^q, l \in Lab^q} rc_{tpl} \ge 1 \quad p \in Ex^q, q \in Q^m, m \in \mathcal{M} \tag{33}$$

$$\sum_{t \in Lec^q, p \in Ex^q} rc_{tpl} \ge 1 \quad l \in Lab^q, q \in Q^m, m \in \mathcal{M} \tag{34}$$

$$\sum_{v \in V^t} y_{tvs} + \sum_{v \in V^p} y_{pvs} + \sum_{v \in V^l} y_{lvs} \le 3 - 2rc_{tpl} \quad t \in Lec^q, p \in Ex^q, l \in Lab^q, q \in Q^m, m \in \mathcal{M}, s \in S^t \cap S^p \cap S^l \tag{35}$$

Constraints (31) establish that if a lecture class is scheduled, then its related exercise and lab classes are scheduled as well. Constraints (32)–(34) force that each class must be related to at least one pair of classes of the remaining types. Constraints (35) prevent related classes from overlapping in the same time-slot.

With these new sets of variables and constraints, we now formulate the ILP model for the first stage. Since the idea is to use its outcome as an input for the second one, mainly setting the assignment of classes to time-slots, we consider an alternative objective function: a weighted combination between maximizing the number of lecture classes assigned and the objective function (23) by considering non-negative weights γ, δ, respectively. As we mention before, each feasible pattern contains one lecture class; moreover, the new inequalities force the assignment of the other type of classes to consist in feasible patterns for the students. We next show the formulation, named GPECTP-RED. Recall that constraints and variables involving room assignment are discarded at this stage:

$$\text{(GPECTP – RED)} \quad \max \gamma \sum_{c \in T^m} w_c + \delta z_{ILP}$$

$$\text{s. t. :} \quad (1)\text{–}(10), (15)\text{–}(22), (24)\text{–}(35)$$

$$\text{restricted to set } \bar{A}$$

$$x_{ac}, xs_{acs}, y_{cvs}, z_{fd}, w_c, rc_{c_1c_2}, rc_{tpl} \in \{0, 1\} \tag{36}$$

Finally, we make some comments regarding this stage. It should be noted that, since the final timetable in the second stage is based on the assignment of classes to time-slots obtained in the first one, it may not be possible to satisfy the minimum student's demand. Intuitively, the definition of $\bar{A}$ may have an impact on the quality of the output generated in the first stage. The larger the number of students included in $\bar{A}$, the better the quality of the resulting assignments. On the other hand, the computation time and memory requirements increase as well.

4.3. Stage 2: Student-class assignment

Given a timetable consisting of the assignment of classes to time-slots, the second stage focuses on obtaining a complete feasible solution by assigning students to courses, as well as determining which room is reserved for each event in a class. We remark that these two aspects are not completely covered in the first stage. Regarding the students, only a reduced subset is considered, and the assignment of classes to rooms is relaxed to simplify the problem. Formally, let $\pi^* = (x^*, xs^*, y^*, z^*, w^*, rc^*, rc^*)$ be the solution vector from the first stage representing the (feasible) class-time-slot timetable. Then, the second stage consists in formulating the model GPECTP-ILP and setting variables $y = y^*$, $w = w^*$ and $z = z^*$.

The second stage can be stopped by two different criteria. Firstly, we consider a time limit, specified by the user, for the execution time on each run. Secondly, once we have at least one feasible timetable, we use the upper bound provided by the LP relaxation of the formulation to abort the optimization whenever it is worse than the objective value of the best solution found. This allows us to avoid solving unnecessary ILPs if the current solution cannot be improved.

5. Computational results and case description

In order to evaluate the behavior of the algorithm, we conducted different experiments at the early stages of the development using artificial instances generated by the university administrative staff. In addition, by the end of 2012, the final tuning and evaluation of the algorithm had been conducted using the information provided by both the students and the faculty staff to construct the timetable for the first semester of 2013. We report the results obtained in the latter, as well as the instances corresponding to the second semester of 2014 and the first semester of 2015, named 1-2013, 2-2014 and 1-2015, respectively. Indeed, we noted that real instances are much harder to solve compared to the artificial one. The experiments were run on a workstation with an Intel(R) Core(TM) i7 CPU (3.40 GHz) and 16 Gb of RAM. The algorithms are coded in C++ using CPLEX 12.1 Callable Library as LP and MIP solver.

Table 1 reports a summary including some characteristics of the instance, such as the number of students ($|\mathcal{A}|$), the average number of courses per preference list (av $|M^q|$), the total number of courses ($|\mathcal{M}|$) and the corresponding number of classes ($|C|$). We also provide the total number of time-slots defined for a week ($|S|$) as well as the number of professors ($|\mathcal{F}|$) and the average number of available time-slots per professor (av $|S^f|$), among other parameters. Finally, in the last two columns we report the number of variables (vars) and constraints (const) in the formulation GPECTP-

Table 1
Description of the instances 1-2013, 2-2014 and 1-2015.

Inst	$\lvert\mathcal{A}\rvert$	av $\bar{m}^a$	av $\lvert M^q\rvert$	$\lvert\mathcal{M}\rvert$	$\sum_{m\in\mathcal{M}}\lvert Q^m\rvert$	$\lvert C\rvert$	$\sum_{c\in C}\lvert V^c\rvert$	$\lvert S\rvert$	$\lvert\mathcal{F}\rvert$	av $\lvert S^f\rvert$	$\lvert\mathcal{R}\rvert$	vars	const
1-2013	981	4.15	7	118	131	341	473	30	220	7.55	31	310 063	751 086
2-2014	915	4.24	6.88	122	136	335	461	36	211	8.66	31	259 395	769 712
1-2015	1065	4.12	6.31	111	118	329	450	36	216	9.5	31	303 229	804 693

ILP. Regarding the different weights involved in the algorithm, we set for the experiments, unless otherwise stated, α, β, γ and δ to one. Although some preliminary experiments were conducted with different values, they are mainly considered to provide the scheduler with the possibility to generate different scenarios during the process.

Concerning the remaining parameters, the student ranking rk^a is a value between 1 and 10 provided by the university and all students assign similar preference values to courses, relative to the order in the preference list. In both cases, a higher value represents a higher ranking and preference. The configuration of the solver is the default one. Regarding the algorithm, a time limit of 1000 s is imposed to each generation of the initial assignment of classes to time-slots (Step 5 in Algorithm 1).

For each of these real-world instances,[1] we conducted the following experiment to evaluate the effectiveness and the quality of the solutions generated. We executed the algorithm considering $l = 0, 5, 10, 20$ and $K = 0\%$ (i.e., students assigned to compulsory courses), $K = 5\%$ and $K = 10\%$. The results are presented in Tables 2–4. For each combination we report the number of feasible solutions found ($|\Delta|$), the objective function and the number of assignments in the best solution found, with their corresponding optimality gaps, and the overall computing time required (in seconds).

To compute the gaps, let z_{feas} be the objective value of a feasible solution and z_{LP} the value of the LP relaxation of GPECTP-ILP. Then, the optimality gap is computed as $\%\text{Gap} = 100*(z_{LP} - z_{\text{feas}})/z_{LP}$. Regarding the number of assignments, the gaps are computed in a similar fashion. The only difference relies in that z_{feas} stands for the number of assignments of students to courses (that is, $z_{\text{feas}} = \sum_{a\in\mathcal{A}} \sum_{m\in M^a} \sum_{c\in T^m} x_{ac}$) and z_{LP} is the value of the LP relaxation of a version of GPECTP-ILP that maximizes this objective. This can be done by setting $\alpha = 0$, $\beta = 1$ and the preference values $wc^{ac} = 1$ for all $a \in \mathcal{A}$, $m \in M^a$, $c \in T^m$, and solving the corresponding LP.

In addition, whenever the required memory in the first stage exceeds the available memory no results are reported for that combination of parameters.

The results obtained are as expected, where larger values of K and l produce in general better results. From the results corresponding to the instance 1-2013 we note that small values of l (5,10) generate large reduced sets $\bar{A}$, and therefore the model GPECTP-RED becomes very difficult to solve and exceeds the available memory. The particular case where $l=0$ does not consider other students than the ones specified in 1 and 2, and therefore the first stage can be solved in this case. For $l=20$ we observe the best results, both in terms of the objective function and the number of assignments. This is justified by the fact that more initial timetables are constructed in the first stage for the reduced sets of students considered as input, and by the fact that each of these problems can be solved with the proposed strategy. Regarding the values of K, we also note an increment in the quality of the solutions as K increases, but at a higher computational effort. Finally, we remark that for $K = 10\%$ the first stage does not finish for any of the values of l since it exceeds the memory limit. A similar tendency is observed for instances 2-2014 and 1-2015, where increasing values of l and K produce the best results, except for the case $l = 0, K = 20$ where the solution found is slightly worse than the one having $K = 5\%$. However, for these two instances we observe that the heuristic is able to find solutions for some combinations of l and $K = 10\%$, since it is possible to solve the first stage for $l=0$ and $l=20$. This difference may rely on the fact that the average number of available time-slots for the faculty staff is greater compared with the instance 1-2013 in both cases, as shown in Table 1.

In order to get a deeper insight into the behavior of the algorithm and the quality of the resulting timetable, we conducted a further experiment using instance 1-2013 to test the impact of increasing the threshold K. However, since the instance is quite tight regarding the set of feasible time-slots for each class, we decided to slightly modify the instance by setting $S^f = \mathcal{S}$ for all $f \in \mathcal{F}$, i.e., professors do not impose constraints for the assignments of classes to time-slots. We also consider $l=0$ for the parameter settings, considering only the students detailed in 1 and 2, since the impact of the value of l has been already studied.

The results are shown in Table 5. For each value of K, we report the execution time (in seconds) in Stage 1. For Stage 2, for each feasible solution in Δ we report its corresponding value in terms of the objective function (23) and the number of assignments, both with their corresponding gaps. We also report the overall execution time required to find the solution.

As expected, we observed that an increase in the value of K produces better final solutions, although the overall computational time increases as well. For $K = 0\%$, the gap of the final solution regarding the objective function (23) is 7.14% and the overall execution time is 2520 s. For $K = 10\%$, these values are 3.74% and 7920. It should be noted that the difference in the execution time relies on mainly in the first phase, where for $K = 0\%$ it requires 3.33 s and for $K = 10\%$ it increases to 5653 s. This increment in the execution times has also an impact on the behavior of the second stage. This is observed in both the number and the quality of the solutions generated, as shown in Fig. 2(a). Restricting the analysis to the execution time of the second stage, higher K values imply the procedure is able to find better solutions in less time. For example, for $K = 10\%$ the procedure finds a solution within a 4.07% gap in 287 s, while for $K = 0\%$ it requires nearly 960 s to find a solution with an 8.70% gap. This behavior is clearly justified by the fact that the assignments generated during the first stage tend to be better as K increases, at the expense of a higher computation time.

We next analyze the quality of the best solution obtained for each value of K. For each student $a \in \mathcal{A}$, we consider his/her *satisfaction value* as the sum of the weights of the courses enrolled in, i.e.,

$$sv_a = \sum_{m\in M^a} w^{am} x_{am}. \tag{37}$$

In addition, we considered an upper bound on this value for each student $a \in \mathcal{A}$, $\bar{sv}_a$, computed by the LP relaxation of the formulation obtained replacing the objective function (23) of GPECTP-ILP by (37). Students usually have preference lists of different sizes, with different preference values for each course. These two definitions allow us to normalize the satisfaction value for each student and compare the level of satisfaction among them. In Fig. 2(b) we report the five-number summary considering the ratio $\frac{sv_a}{\bar{sv}_a}$ for all $a \in \mathcal{A}$, ordered by their ranking weight rk^a and partitioned into three subsets of equal size. We remark that for students having all of their courses marked as mandatory this value is considered as 1.0, and that the three sets are defined in the same way for the three values of K considered. From a general perspective, we can observe that the satisfaction level increases when K is larger, with higher median values and less dispersion of the satisfaction ratios. This is consistent with the results in Table 5, where the objective values for higher values of K increase. In addition, as K increases the satisfaction values tend to vary according to the students rankings. For example, for $K = 10\%$ the dispersion of each group tends to be larger for low ranking students, while for

[1] Instances can be retrieved from http://www.dc.uba.ar/Members/pzabala/GPECTP-instances.tar.gz.

Table 2
Objective function and computational time required in a real-world instance for different K and l values, 1-2013.

l/K	0%						5%					
	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time
0	2	50 514	6.61	3898	4.01	1680	3	51 421	4.95	3929	3.25	1500
5	ml	–	–	–	–	–	ml	–	–	–	–	–
10	ml	–	–	–	–	–	1	52 449	3.03	3999	1.53	63 902
20	8	52 232	3.44	3979	2.01	8580	9	52 457	3.02	3997	1.58	67 500

$K = 0\%$ there is no clear pattern. This behavior is justified by the construction of the timetables in stage one, where increasing the value of K provides additional information to the algorithm including students' requirements which are not necessarily mandatory.

We conducted two other experiments to provide further evidence and a deeper insight into the behavior of the approach, mainly regarding the impact of the decision made for the construction of the timetables in the first stage. Recall that the timetables generated in the first stage are driven by the reduced set of students in $\bar{A}$. The objective function defined in GPECTP-RED considers partially their preferences and the courses selected by them. For each instance, we consider three different configurations for the algorithm:

- Conf-base: $l=0$ and $K=0$, and therefore the first stage only considers the required subjects. These are the values reported in Tables 2–4**?**, which are used as a baseline.
- Conf-no-prefs: we take the best configuration obtained for each instance, and set $\delta = 0$. As a result, the objective function of GPECTP-RED maximizes the assignments of classes to time-slots. The assignment of mandatory course and the minimum number of courses assigned to the top $K\%$ are guaranteed by the constraints in the model. The configurations are $l=20$, $\delta = 0$, $K = 5\%$ for instance 1-2013 and $K = 10\%$ for instances 2-2014 and 1-2015.
- Conf-best: the best configuration regarding the results in Tables 2, 3 and 4, including the preferences of the students. The same as Conf-no-prefs but setting $\delta = 1$.

The results obtained are shown in Table 6. We observe that the overall results of the best solution found, as well as the number of assignments, are improved by including the preferences of the (reduced set) of students in the first stage. In particular, this can be observed when comparing Conf-no-prefs with Conf-best, since the only difference between them is the objective function of GPECTP-RED.

In general, many algorithms in the related literature (see, e.g., [22,10] where classes are assigned to days of the week) propose a first stage consisting of the assignment of classes to time-slots which, once fixed, are used as input in a subsequent procedure to assign rooms and students. However, our proposal performs an initial assignment using partial information of the second stage, that is, conditioning the assignment by using a reduced set of students (those with required subjects plus the top $K\%$) to influence the outcome. In addition, by considering l different reduced student sets $\bar{A}$, the multi-start nature of the approach increases the chances to find better overall solutions.

Finally, we aim to evaluate the impact of the students rankings as well as the preferences established for the courses in the algorithm. Therefore, we generated a scenario where the objective function accounts for neither the students rankings nor their course preferences. The results are shown in Table 7, where we report the overall objective function and the computation time required compared to the best solution found in the standard case. The objective function reported in both cases considers the preferences and the students ranking. In those cases where no preference was selected, we compute these values using the information provided in the original instance.

The results show that considering the priorities has an impact on the solution, increasing the overall satisfaction value by nearly 9–10%. This means that allowing the students to provide a preference list with their corresponding weights has a positive impact regarding the overall solution and the selection of the students. Nevertheless, the computational times show an increment on the three instances.

Finally, we make some comments regarding the outcome of the project at an organizational level. The algorithm has been implemented and used to construct the timetables since the first semester of 2013, and it is still being used at the beginning of each semester. From the organizational standpoint, the implementation of this tool produced several benefits. The process of constructing the timetable has been reduced from two months to just a few days, and the scheduler is now able to consider different scenarios (for example, with faculty staff availability) to select the best timetable according to their needs. In addition, the students now have a higher level of freedom to decide how to organize their semester and program of study, being able to include their preferences in the course assignment. Since prior to this new policy enrollment was compulsory, the requests for modifications to the assignments at the beginning of the semester have been considerably reduced, allowing the administrative staff to focus on more important adjustments (operational, last minute availability changes) to the timetable. In general, the first feasible solution is obtained after 30 min of execution, and on average approximately 10 solutions are reported after 3 h. A complete run of the algorithm takes usually about 10 h, and is in general run overnight.

Table 3
Objective function and computational time required in a real-world instance for different values of K and l, 2-2014.

l/K	0%						5%						10%					
	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time
0	3	46 199	9.44	3635	6.25	1245	4	48 588	4.75	3750	3.02	1457	5	48 657	4.62	3766	2.61	1977
5	ml	–	–	–	–	–	ml	–	–	–	–	–	ml	–	–	–	–	–
10	3	48 926	4.09	3773	2.43	19 324	ml	–	–	–	–	–	ml	–	–	–	–	–
20	13	49 059	3.83	3769	2.53	11 694	8	49 281	3.4	3792	1.93	35 575	8	49 356	3.25	3790	1.99	81 934

Table 4
Objective function and computational time required in a real-world instance for different values of K and l, 1-2015.

l/K	0%						5%						10%					
	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time	$\|\Delta\|$	Obj.	%Gap	#Assign.	%Gap	Time
0	2	45 321	10.84	3935	8.68	9159	3	46 391	8.74	4039	6.27	3099	4	46 021	9.47	3994	7.31	9661
5	ml	–	–	–	–	–	ml	–	–	–	–	–	ml	–	–	–	–	–
10	3	47 040	7.46	4065	5.66	56 214	2	46 946	7.65	4052	5.96	66 772	ml	–	–	–	–	–
20	5	47 366	6.82	4104	4.76	77 646	11	47 482	6.59	4119	4.41	77 575	8	47 833	5.9	4140	3.92	102 042

Table 5
Objective function and computational time required in a real-world instance 1-2013 for different values of K, $l=20$.

Stage 1	0% Time 3.33					5% Time 676					10% Time 5653				
Stage 2	Obj.	%Gap	#Assign.	%Gap	Time	Obj.	%Gap	#Assign.	%Gap	Time	Obj.	%Gap	#Assign.	%Gap	Time
1	49 383	8.70	3846	5.29	960	51 476	4.83	3948	2.78	1200	51 888	4.07	3974	2.14	5940
2	49 869	7.81	3870	4.70	1080	51 805	4.23	3963	2.41	1440	51 986	3.89	3979	2.01	6000
3	50 231	7.13	3894	4.11	1440	51 842	4.16	3954	2.63	1740	52 003	3.86	3980	1.99	6060
4						51 921	4.01	3966	2.33	1860	52 015	3.84	3979	2.02	6120
5											52 067	3.74	3975	2.12	6900
Total time					2520					2820					7920

6. Conclusions and future directions

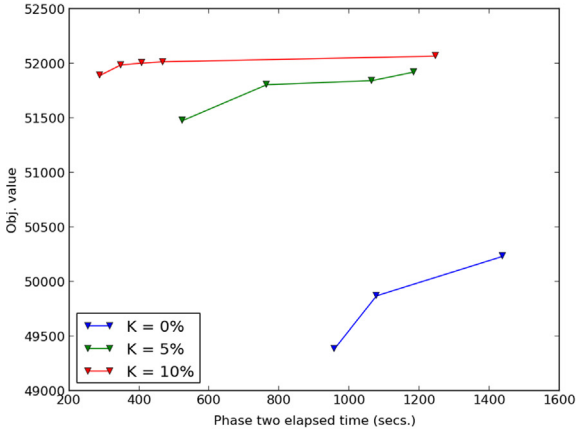
In this paper we study the Generalized Post-Enrollment Course Timetabling Problem (GPECTP), a new problem that generalizes two known problems from the related literature, motivated by a real-world application in the university context. We propose an ILP formulation for this problem which models all constraints related to the selections made by students as well as the operational constraints imposed by the university, and develop a two stage heuristic solution approach based on this mathematical formulation.

The heuristic has been tested on three real-world instances, obtaining good quality results that were used in practice to schedule the semester. The approach is able to find solutions within a 3–4% quality gap while satisfying nearly a similar percentage of the students' requests. The proposed solution has been deployed and used to schedule each semester since the first semester of 2013.

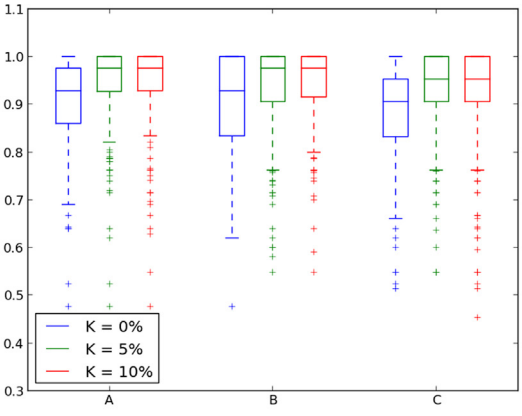
As future research, we first make some remarks about possible extensions of GPECTP-ILP. Some of them have already been incorporated into the formulation and the heuristic without major changes despite some extra definitions. These extensions are:

- *Set of feasible time-slots for each room*: During some time-slots a subset of the rooms are not available since they are reserved for other academic activities. These feature has been incorporated into the formulation by restricting the assignment of events to rooms in some particular time-slots.
- *Reserved number of places per class*: the university has blocked a certain number of places per class in case of unforeseen events. It will therefore be possible for the university to make changes to the timetable. In addition, this fixed number of places is useful for first-year courses in each program of study, where the demand for them is estimated based on historical records and variations in these estimations may occur.

Finally, regarding the computational aspects of the approach, it would also be interesting to derive valid inequalities for GPECTP-



(a) Quality of solutions found during Phase 2



(b) Students' satisfaction boxplot, divided in three groups and sorted by ranking priority in descending order

Fig. 2. Graphic overview of the solutions found by the heuristic for each value of K. (a) Quality of solutions found during Phase 2. (b) Students' satisfaction boxplot divided in three groups and sorted by ranking priority in descending order.

Table 6
Results corresponding to the three instances with different settings for the first stage.

Inst	Conf-base			Conf-no-prefs			Conf-best		
	$\|\Delta\|$	Obj.	Time	$\|\Delta\|$	Obj.	Time	$\|\Delta\|$	Obj.	Time
1-2013	2	50 514	1680	15	50 843	15 180	9	52 457	67 500
2-2014	3	46 199	1245	10	48 054	18 360	8	49 356	81 934
1-2015	2	45 321	9159	5	46 615	31 200	8	47 833	102 042

Table 7
Results for the three instances varying the students' preferences.

Inst	l	K	w/o prefs.		With prefs.	
			Obj.	Time	Obj.	Time
1-2013	20	5	48 435	14 700	52 457	67 500
2-2014	20	10	45 597	18 660	49 356	81 934
1-2015	20	10	44 460	49 020	47 833	102 042

ILP, in order to evaluate whether it can be used in practice. Moreover, some of them may be included in the heuristic as well, and further inequalities can be derived for the formulation considered in the first stage, aiming to reduce the overall computational time of the approach.

Acknowledgments

The authors are grateful to three anonymous referees and the associate editor for their valuable comments and suggestions for improving this paper. They also thank the authorities of the university for the permission to publish the information from the instances.

This research is partially supported by grants PICT-2010-0304, PICT-2011-0817 and UBACyT 20020100100666.

References

[1] Bettinelli Andrea, Cacchiani Valentina, Roberti Roberto, Toth Paolo. An overview of curriculum-based course timetabling. TOP 2015;23(2):313–49.
[2] Burke Edmund K, Marecek Jakub, Parkes Andrew J, Rudov Hana. A branch-and-cut procedure for the eudine course timetabling problem. Ann Oper Res 2012;194(1):71–87.
[3] Burke Edmund K, Pham Nam, Qu Rong, Yellen Jay. Linear combinations of heuristics for examination timetabling. Ann Oper Res 2012;194(1):89–109.
[4] Cambazard Hadrien, Hebrard Emmanuel, O'Sullivan Barry, Papadopoulos Alexandre. Local search and constraint programming for the post enrolment-based course timetabling problem. Ann Oper Res 2012;194(1):111–35.
[5] Michael W. Carter, A comprehensive course timetabling and student scheduling system at the University of Waterloo. In: Practice and theory of automated timetabling III; 2001. pp. 64–82.
[6] Ceschia Sara, Di Gaspero Luca, Schaerf Andrea. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. Comput Oper Res 2011;39(7):1615–24.
[7] Ceschia Sara, Di Gaspero Luca, Schaerf Andrea. The generalized balanced academic curriculum problem with heterogeneous classes. Ann Oper Res 2014;218(1):147–63.
[8] Chiarandini Marco, Di Gaspero Luca, Gualandi Stefano, Schaerf Andrea. The balanced academic curriculum problem revisited. J Heurist 2012;18(1):119–48.
[9] Marco Chiarandini, Chris Fawcett, Holger H. Hoos, A modular multiphase heuristic solver for post enrolment course timetabling. In: Proceedings of PATAT; 2008.
[10] Daskalaki S, Birbas T. Efficient solutions for a university timetabling problem through integer programming. Eur J Oper Res 2005;160(1):106–20.
[11] Gogos Christos, Alefragis Panayiotis, Housos Efthymios. An improved multi-staged algorithmic process for the solution of the examination timetabling problem. Ann Oper Res 2012;194(1):203–21.
[12] Jat Sadaf Naseem, Yang Shengxiang. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. J Sched 2011;14 (6):617–37.
[13] Kristiansen Simon, Sørensen Matias, Stidsen Thomas R. Integer programming for the generalized high school timetabling problem. J Sched 2014:1–16 (English).
[14] Lach Gerald, Lübbecke Marco E. Curriculum based course timetabling: new solutions to udine benchmark instances. Ann Oper Res 2012;194(1):255–72.
[15] Lewis Rhyd. A time-dependent metaheuristic algorithm for post enrolment-based course timetabling. Ann Oper Res 2012;194(1):273–89.
[16] Lu Zhipeng, Hao Jin-Kao. Adaptive tabu search for course timetabling. Eur J Oper Res 2010;200(1):235–44.
[17] Lübbecke Marco E. Comments on: an overview of curriculum-based course timetabling. TOP 2015;23(2):359–61.
[18] McCollum Barry, Schaerf Andrea, Paechter Ben, McMullan Paul, Lewis Rhyd, Parkes Andrew J, et al. Setting the research agenda in automated timetabling: the second international timetabling competition. INFORMS J Comput 2010;22 (1):120–30.
[19] Müller Tomáš, Murray Keith. Comprehensive approach to student sectioning. Ann Oper Res 2010;181(1):249–69.
[20] Keith Murray, Tomáš Müller, Hana Rudová, Modeling and solution of a complex university course timetabling problem. In: Practice and theory of automated timetabling VI; 2007. pp. 189–209.
[21] Nothegger Clemens, Mayer Alfred, Chwatal Andreas, Raidl Günther R. Solving the post enrolment course timetabling problem by ant colony optimization. Ann Oper Res 2012;194(1):325–39.
[22] Sørensen Matias, Dahms Florian HW. A two-stage decomposition of high school timetabling applied to cases in Denmark. Comput Oper Res 2014;43 (0):36–49.
[23] van den Broek John, Hurkens Cor. An ip-based heuristic for the post enrolment course timetabling problem of the itc2007. Ann Oper Res 2012;194(1):439–54.
[24] van den Broek John, Hurkens Cor, Woeginger Gerhard. Timetabling problems at the tu Eindhoven. Eur J Oper Res 2009;196(3):877–85.