UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

ENGINEERING SCHOOL

DEPARTMENT OF INDUSTRIAL AND MECHANICAL ENGINEERING

CONSTRAINT AND INTEGER PROGRAMMING APPROACHES FOR THE SOLUTION OF A COURSE TIMETABLING PROBLEM.

THESIS PRESENTED TO FULFILL THE REQUIREMENTS OF THE HONORS PROGRAM BY THE STUDENT

DIANA ANGÉLICA VÁZQUEZ LIMÓN

155967

DIRECTOR

DRA. DOLORES EDWIGES LUNA REYES

SAN ANDRÉS CHOLULA, PUEBLA. FALL, 2020

THESIS PRESENTED TO FULFILL THE REQUIREMENTS OF THE HONORS PROGRAMM BY THE STUDENT DIANA ANGÉLICA VÁZQUEZ LIMÓN, 155967

THESIS DIRECTOR

Dr. Dolores Edwiges Luna Reyes

THESIS PRESIDENT

Dr. Juan Antonio Díaz García

THESIS SECRETARY

Dr. Nelly Monserrat Hernández González

Thanks to my mentor Dr. Dolores Luna, for all your time and knowledge you have given me. Also I am thankful for my colleagues María José Magallanes and Israel Jaser Gómez for being at the beginning of the development of this work. I dedicate this work to my family and friends who have encouraged me and have always been there for me.

Contents

1 Introduction **5**

2 Literature Review **7**

3 Problem Definition **10**

4 Methodology **12**
4.1 Integer Programming 12
4.2 Constraint Programming 13

5 Results **16**
5.1 One-section courses Integer Programming examples. 16
5.2 Multiple-section courses Integer Programming example. 25
5.3 Constraint Programming example. 28
5.4 Instances solutions with Integer programming. 29
5.5 Instances solutions with Constraint programming. 45

6 Conclusions **60**

References **62**

Abstract

The timetabling problem is a well-known problem, when it is related to the academic environment it seeks to assign courses into schedules considering teaching staff, students, classrooms and requirements of the courses. This problem is addressed at least twice a year in our university. Each case of study has its' own requirements and constraints, and manual solution of this problem may be exhausting and time consuming. Therefore, looking for a way to obtain better assignations and to ease the workload of the task, many researchers have studied it. For this work, we faced the case of assigning the courses to a schedule having as data the teachers assigned to the courses and the number of groups needed. The assignation of courses to classrooms was not a query for the problem. The interest was in having no overlaps among the courses taught by the same teacher nor the courses that students were meant to study in the same semester. Two approaches are presented, the first involves an integer programming model with binary variables and a set of constraints to ensured the requirements are met. The second one is a constraint programming model that uses constraints to delimit the domains of variables and to state the requirements needed for the assignation. Each of these solutions was modeled and evaluated in two instances to be compared to what the real-life assignation realized by a person was.

Key words: timetabling, course scheduling, integer programming, constraint programming.

1 Introduction

We have all been subject to a schedule in any time of our life, either it was for academic reasons or for work. But, have you ever wondered, how does the schedules are created? or more over, who is in charge of making those schedules? This is a query not everyone is aware of. The creation of schedules is part of a problem faced in many fields, being the academic the one we are interested in this work. This problem is denominated timetabling problem and is widely used. At our field of interest, academic, we have different schedules each semester, that means that this problem is faced at least two times a year. The assignation of schedules to a list of courses may be a long and tiring task, and the people in charge of doing it may be doing it by empirical methods which result in fatigue and can be a cause to human errors of wrong assignation.

Recent studies approach the problem with different methodologies, some of which are combinations of different methods. Each of the studies in the literature makes different assumptions for the problem. Some of them have the need to assign together with the courses to a schedule, teachers or classrooms, others face the problem of also assigning groups of students. As the number of elements to be considered in the timetabling problem increases, the difficulty of the problem under study also increases and its solutions increase in complexity.

As a first approach, a complex problem can be approximated by simplifying some of the assumptions and thus making the problem easier to solve, in terms of less computational time and more options to choose from. Furthermore, in reality many times a problem is broken down and resolved in stages. For example, you can first make the assignment of teachers to courses and then create the course schedule, why would you try to use a solution that also seeks to assign teachers when you have already established that assignment? It would be absurd and it would consume a lot of resources.

In the environment in which this study was conducted, the study instances make precisely the assumption that the assignment of teachers to courses is done first. Subsequently, the courses are assigned to a schedule. Therefore, to find a solution to a scheduling problem, it would be wise to first study and identify the situation in which it is needed.

Due to the approach described above, we propose to solve the problem under study using a model that helps in the allocation of schedules that is simple and complies with what is needed for our instances. We want a model that covers what is necessary and is functional. Knowing this, we also need to know the situation in which the model will be used and what the requirements for the model will be. In this work, we intend to find a solution proposal to the problem of schedules applied in a particular case of course assignment within a university department in charge of three undergraduate programs. The methodologies used in this work are an integer programming model using binary variables for the assignment, and a constraint programming model that is solved using the Kalis Xpress library.

The rest of this document is organized as follows. In first place, we present some of the previous studies carried out on the timetabling problem related to the problem studied in this work. Next, we present the definition of our situation. Subsequently, we describe the proposed methodologies for the solution, followed by a set of examples that explain how these methodologies work. Also, we show the application of the proposed models

with the solutions to two instances of past semesters courses that will be compared with the assignment made in those circumstances. Finally, we present some conclusions.

2 Literature Review

Timetabling problem is a problem that has been approached many times in the literature as it has applications in many fields such as health care (Cardoen, Demeulemeester, & Beliën, 2010), academic, industrial environments, and more. For that reason, aside from having many research projects on this topic we also have compilations of what has been worked into it.

To begin, we shall know what is understood under the term timetabling problem, Abdullah, Burke, and McCollum refers to timetabling problem as the assignment of a set of events into specific time slots and rooms within a determinate time frame, taking into account soft and hard constraints (Abdullah et al., 2007). Saldaña Crovo, Oliva San Martín, and Pradenas Rojas say timetabling consists of programming courses, taking into account teaching faculty, days, slots, classrooms and other requirements for the courses (Saldaña Crovo et al., 2007) and Lach and Lübbecke consider timetabling as the conflict-free assignment of events to time slots and rooms, having hard and soft constraints to be fulfilled (Lach & Lübbecke, 2008). From this, we can come up with considering timetabling as the problem in which we have events we want to assign to a determined time, space and person to be involved in it. For this, we will have a set of constraints that will dictate how this assignation has to be done. In our particular case, we are interested into the timetabling problem for a university courses assignation (academic environment).

Babaei, Karimpour, and Hadidi and Teoh, Wibowo, and Ngadiman with their respective teams have worked into analyzing the approaches that have been done to propose feasible solutions for the problem. Babaei et al. begin giving their definition for the problem, they call it the "University Course Timetabling Problem" (UCTTP) and define it as a NP-hard problem which involves the allocation of events to a fixed number of time slots and classrooms. They consider events to be courses, teachers and students (Babaei et al., 2015). As for Teoh et al., they call the problem "Academic scheduling problems", considering it a NP-hard and NP-complete problem. They mention that computational time increases exponentially with the size of the problem (Teoh et al., 2015).

The approaches these authors cover in their literature reviews are: operational research methods, meta-heuristics and hybrid algorithms. Babaei et al. mention within the operational research techniques used for solving this problem Integer Programming/Linear Programming methods, Constraint Satisfaction Programming and Graph Coloring theory. For meta-heuristics the authors present approaches using population based metaheuristics, evolutionary and genetic algorithms, ant colony optimization algorithm, memetic algorithm, single solution based metaheuristic, tabu search, simulated annealing, local search, Variable Neighborhood Searching (VNS), and Fuzzy approach (Babaei et al., 2015).

Teoh et al. present on their literature review techniques such as graph colouring, integer and linear programming and, dynamic programming, but they find all these approaches not practical and inflexible. Therefore, they focus their literature review on meta-heuristic techniques. In their review, they divide all works in three categories: one-stage optimization (which work simultaneously hard and soft constraints), two-stage optimization (first works with the hard constraints and after that works with the soft constraints) and algorithms that allow relaxation (similar to two-stage optimization but

here, we allow a relaxation on the first stage). The meta-heuristics they mention on their review are Tabu search, hyper-heuristics, genetic algorithm, simulated annealing, ant colony optimization, the great Deluge and particle swarm optimization. They also have another category for hybrid algorithms in which he considers fuzzy genetic, two point hybrid and evolutionary algorithm (Teoh et al., 2015). As we can see, both papers mention similar approaches, meaning these are the more significant.

There is always mentioned that the timetabling problems need to have hard and soft constraints. Babaei et al. define hard constraints as those that under any condition mustn't be violated and soft constraints are those that shouldn't be violated as much as possible. In most of the approaches hard constraints are used to delimit the feasible area for solution and the soft constraints are for the objective function being the aim to minimize the number of soft constraint that are violated.

Some examples for hard and soft constraints are mentioned by Babaei et al. Examples of hard constraints are: a teacher cannot attend two classes at the same time; a course is taught by only one teacher, in one room and with only one group of students for each time slot; the capacity of the classrooms should be proportional to the size of the group of students. Examples for soft constraints are: preferences of the teacher to certain time slots, preferences on classrooms, minimization of empty time slots for teachers and/or students, lunch break, occupation of days, etc. This constraints may vary between instances because maybe what is considered by Babaei et al. as a soft constraint may be considered as a hard constraint for a particular institution.

We find integer programming approaches in Daskalaki, Birbas, and Housos (Daskalaki et al., 2004), Hernández Campos (Hernández Campos, 2008), Schimmelpfeng and Helber (Schimmelpfeng & Helber, 2007), MirHassani (MirHassani, 2006), Lach and Lübbecke (Lach & Lübbecke, 2008), and Saldaña Crovo et al. (Saldaña Crovo et al., 2007). They all worked with an integer programming approach using binary variables, considering hard and soft constraints using as objective function the minimization of the number of soft constraints not fulfilled. They all got good results but with high computational times.

Daskalaki et al. work with multidimensional variables, which give flexibility to the model for it to comply with more specific rules that were turned into mathematical constraints (Daskalaki et al., 2004). For the version of the problem they studied, it is needed to assign courses to classrooms and slots and teachers. They are working with a large set of constraints which for their instance worked fine, but when the instance grew in size, the computational time increased, so later on, they worked on a efficient solution to that. This new approach involved a two-stage relaxation, the main idea was to first assign the courses to a day and on the next stage to rearrange the courses per day. This worked out, obtaining a significant reduction in the computational time (Daskalaki & Birbas, 2005).

Hernández Campos proposes three different approaches for using integer programming methodologies. In the first one, he does the direct identification for courses, days, slots, and classrooms, to create decision variables and write all the constraints for the model to resolve the assignation. In his second approach, he divides the model in two stages, in the first stage, he assigns slots and in the second stage, he assigns classrooms. For the third approach, he needs to pre-define all the feasible combinations of slots to work with them as a set and with that solve all the problem in one-stage (Hernández Campos, 2008).

We can see with the description of the two previous works using integer program-

ming, that in general, they are seeking to assign the courses to a classroom and a time slot, having at least the set of courses, slots, teachers and classrooms. In general, they got good results but when trying with big instances they found large computational times which some authors tried in further studies to reduce.

As for other approaches, we found many works using meta-heuristics and hybrid heuristics. See for example, Badoni, Gupta, and Mishra (Badoni et al., 2014), Bolaji, Khader, Al-Betar, and Awadallah (Bolaji et al., 2014), Pongcharoen, Promtet, Yenradee, and Hicks (Pongcharoen et al., 2008), Gunawan, Ng, and Poh (Gunawan et al., 2007) and Abdullah et al. (Abdullah et al., 2007).

Abdullah et al. (Abdullah et al., 2007) work with a neighborhood structure with a randomized iterative improvement algorithm, Pongcharoen et al. (Pongcharoen et al., 2008) present a stochastic optimization timetabling tool which had embedded genetic algorithms, simulated annealing and random search to ensure rectification of infeasible timetables, Gunawan et al. (Gunawan et al., 2007) work on a hybrid algorithm which involved integer programming, a greedy heuristic and a modified simulated annealing algorithm for the assignment of teacher and course scheduling, having satisfactory results but once again with high computational times. Later Gunawan, Ng, and Poh worked on improving the previous work and came up with a hybridized Lagrangian relaxation for the stage with integer programming to then improve the output with a simulated annealing algorithm (Gunawan et al., 2012). As for Badoni et al. work with NHA (new hybrid algorithm) which had a list of events (different types of courses) and had to create mutually disjoint groups of students to do the assignation of groups to courses, to slots, to teachers and to classrooms. They compare the results obtained with their approach to the results obtained with a genetic algorithm with local search and obtained promising results (Badoni et al., 2014). And with Bolaji et al. (Bolaji et al., 2014), we have an approach which worked with a novel hybrid ABC (Artificial Bee Colony) algorithm.

All the previous approaches show good results which lead as Teoh et al. mentions to promote the development of more approaches for the problem (Teoh et al., 2015). Until now, there has not been a specific methodology that comes as the best one as each instance is different and has its' own specifications and for that each approach has also its' own strengths and advantages.

3 Problem Definition

In this section, we define the characteristics of the problem that we study in this work. The problem under study is the assignment of courses to a schedule in the Department of Industrial and Mechanical Engineering. We are faced with the need to schedule each semester the courses of three different bachelor degree programs. The Academic Director analyzes which courses should be offered and how many groups for each course are required given the demand of the students. The director also defines which professor will be in charge of teaching each of the courses. We do not have to worry about the capacity of the classrooms. The problem is to assign a schedule to each course, considering that we want to avoid overlaps between courses that students should take in the same semester and courses taught by the same teacher.

The Scholar Services department establishes the rules for the allocation of schedules that allow efficient use of the institution's resources. These rules are dynamic, and they change as the characteristics of the education in the institution and the environment change.

Every bachelor's degree program consists of eight semesters. At this time, the courses from the first to the fourth semester should be taught between 8 and 15 hours, while the corresponding courses from the fifth to the eighth semester should be scheduled between 13 and 21 hours or only one class at 7 hours. In addition, we have certain structures for each course, this means that the weekly hours to be taught vary according to the course.

In order to model the problem under study, we need to find a way to represent the different structures that exist. We have six different structures, each one indicates the total time a course will be taught in a week, and this time can be distributed throughout the week in 1,2, 3 or 4 days.

- Structure 1: 150 minutes in three, two or one sessions per week.
- Structure 2: 200 minutes in four or two sessions per week.
- Structure 3: 150 minutes in two or one sessions per week.
- Structure 4: 100 minutes in two or one sessions per week.
- Structure 5: 200 minutes in two or one sessions per week.
- Structure 6: 300 minutes in three, two or one sessions per week.

These structures can be scheduled in a set of time slots or blocks appropriate to satisfy the times of each one of the structures. When observing the schedules, we noticed that there are specific schedules to start classes for each structure, so we made the schedules for all the possibilities and then proceeded to number them. This can be seen in Figure 1. So, as we can observe, to schedule a course in block 1 corresponds to scheduling it on Mondays, Wednesdays and Fridays from 7 to 7:50 hours since it is a structure 1 with 50 minutes three times a week, being 150 minutes per week, or that if a course is scheduled in block 261, we will have it on Mondays from 17:00 to 20:20, since it is in structure 5 with 200 minutes in one day and so on.

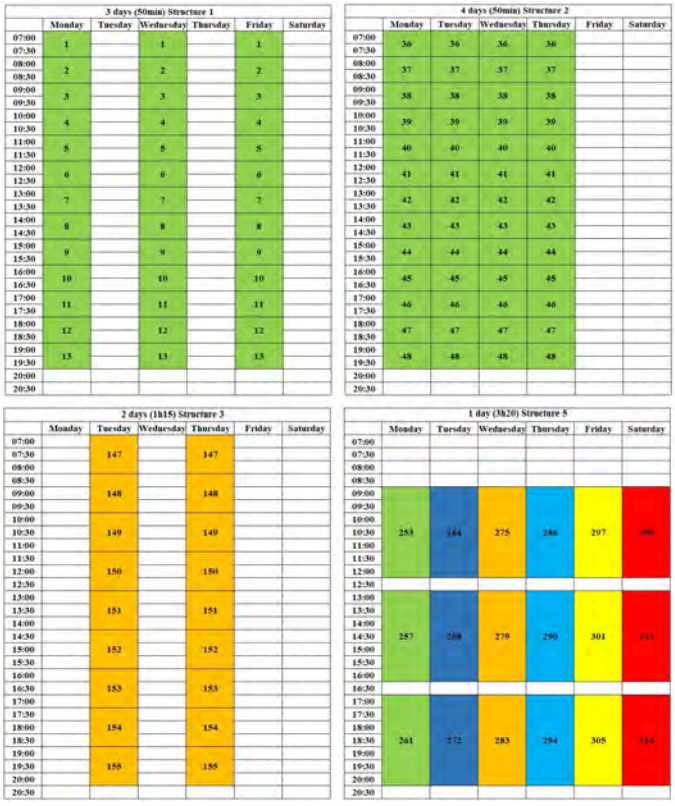


Figure 1: Example of some timetables built to number the possible blocks to schedule, divided by structure.

In this way, we obtained the list for possible time slots or blocks in which we can program our courses. As we have defined all the structures used, we wouldn't need to make this more times as those are expected remain the same for all the semesters, what we change are the courses to be scheduled.

There are several reasons that motivate to find solutions to the problem studied in this work. On one hand, from an administrative point of view, since this is a process that is done every semester, having an automatic scheduling tool will streamline the process by giving the person in charge time to focus on other problems that require their attention. On the other, from the students' point of view, this type of tool will allow us to offer them better schedules.

An additional difficulty in the problem arises because although the students are supposed to go according to the curriculum, there are a significant number of students who do not respect this restriction and take courses of several semesters. Finally, it is also important to mention that two study plans for the same degree program frequently coexist.

4 Methodology

In this section, we describe the proposed methodologies that are used to find solutions to the problem described in the previous section. It is important to mention that the problem studied in this work does not have an objective function, that is, we do not seek to optimize any measure that defines how adequate a course schedule is. In the case under study, any feasible solution, that is, a solution that satisfies all the restrictions, is an adequate solution to the problem. For this reason, it is convenient to generate more than one solution so that the decision maker can select from among the feasible solutions the one that best meets those needs not explicitly included in the model.

We use two different approaches to find feasible solutions to the problem studied in this work: integer programming and constraint programming, both methodologies are described in more detail below.

4.1 Integer Programming

In this section, we describe an integer programming model that can be used to find solutions to the problem. As mentioned in the previous section, the problem is to schedule a set of courses for three different undergraduate programs. For each of the courses the following is previously defined: the number of groups or sections required, the teacher who will teach each section and the structure it has. Given the structure of the course, the blocks or time slots in which it can be programmed are known. With these information, we want to generate course schedules that do not have overlaps of teachers, nor of courses that must be taken in the same semester or term.

Let J be the set of courses to be scheduled, I be the set of available blocks or time slots, S the set of semesters in a degree program and, T the set of teachers. For each semester $s \in S$, lets define JS_s as the set of courses that belong to a semester in an undergraduate program. For each course $j \in J$ lets also define IJ_j as the set of blocks where a course can be assigned and, for each teacher $t \in T$ let K_t be the set of courses assigned to them. Lets finally define the following variables:

$$x_{ij} = \begin{cases} 1 & \text{course } j \text{ is assigned to the block } i \quad \forall i \in I, j \in J \\ 0 & \text{o.w.} \end{cases}$$

The model proposed to solve the problem studied in this work follows:

$$\sum_{j \in JS_s} x_{ij} \leq 1 \quad \forall i \in I, s \in S \tag{1}$$

$$\sum_{i \in IJ_j} x_{ij} = 1 \quad \forall j \in J \tag{2}$$

$$\sum_{j \in K_t} x_{ij} \leq 1 \quad \forall i \in I, t \in T \tag{3}$$

Constraints (1) prevent to schedule two courses from the same semester in the same time slot. Constraints (2) guarantee that all courses are assigned to one time slot. Finally, constraints (3) guarantee that all courses assigned to one teacher do not overlap.

4.2 Constraint Programming

Constraint programming is a paradigm that can be used to solve combinatorial problems that is based on a combination of techniques, from artificial intelligence, computer science, databases, programming languages and operations research. Constraint programming has been used in many applications such as scheduling, planning, vehicle routing, configuration, networking, etc.

The general idea of constraint programming is to represent a problem in terms of decision variables and constraints. The constraint solvers take the problem model, and find an assignment to all the variables that satisfies the constraints. Constraint solvers search the solution space systematically, such as with backtracking or branch and bound algorithms, or they use local search algorithms. The systematic method often intersperses search and inference, where inference consists of propagating the information contained in a constraint to neighboring constraints. Such inference (called constraint propagation) is useful as it can narrow down the search space. While defining a set of constraints may seem like an easy way to model a real-world problem, finding a good model that works well with a chosen solver is not always easy. A poorly chosen model can be very difficult to solve (Rossi, Van Beek, & Walsh, 2006).

For the second approach, we propose to model the problem as a Constraint Programming (CP) problem. To solve the model we use Kalis, the Constraint Solver of XPRESS.

As mentioned above, Constraint Programming works by defining decision variables and their domains to establish constraints over the variables. A CP problem is built by adding constraints and bounds on the variables to find a feasible solution.

For our problem we would need to define our set of courses to be imparted and the set of time slots in which the courses can be scheduled. In terms of our constraints, we need to remember we are working with 6 different structures which are defined by the course and also that we don't want to have any overlaps, this means that we have to pay attention to avoid having courses taught by the same teacher and courses from the same semester scheduled at the same time.

For the model of the problem, we use as a basis a model in the user guide for Kalis XPRESS for a timetabling problem (Xpress, 2014). They had it for small instances and in the simplest version of the problem. Assigning a number of meetings to certain schedules. Considering J the number of meetings and I

the number of possible schedules. Let the decision variable be a set of size J which for values would take a value from 1 to I. As follows:

$$plan_j = 1, 2, ..., I \quad \forall j \in J$$

They worked with a single set of constraints they called "INCOMPATIBILITIES" which stated that if two meetings couldn't be carried on at the same time, the answer to its' schedule must be different. Let j and k represent numbers of meetings that can't be scheduled at the same time in the next equation:

$$plan_j \neq plan_k \quad \forall j \in J, k \in J \mid j \neq k$$

Considering this model, the only thing we are missing to take into account is the structures. In further examples, the model is extended for the case where certain meetings can't take place at a determined schedule. This was solved by first creating the variables $plan$ taking into account that it could take any value from 1 to I but then adding some expressions involving the variable to state that $plan$ couldn't take or needed to be a certain schedule. An example could be, if we know that meeting 1 cannot be on schedule 2 there must be shown on the model as:

$$plan_1 \neq 2$$

As the problem under study is very similar, we could use the previous example as a base for our own problem. We would have more data meaning we needed a way to work with the volume of data we had to avoid having to write each of the incompatibilities one-by-one.

To work our instance using Constraint Programming let J be the set of courses while I is the set of time slots. Defining $plan$ as our variable which is described in equation (4). This was defined based on the example described in the user guide of Kalis, we will also work with the set of constraints called "INCOMPATIBILITIES" which are for stating if the schedules for 2 courses can't be scheduled at the same time slot either because of the teacher being the same or the courses belonging to the same semester, these constraints are shown at equation (5).

$$plan_j \in I \quad \forall j \in J \tag{4}$$

$$plan_j \neq plan_h \quad \forall j, h \in J \text{ and } j \neq h \tag{5}$$

Recalling what was mentioned earlier, when we have small instances, it can be easy to write the constraints one by one but as our instances are not small we can declare an array of size (J, J) to declare the existence of overlaps. This array can be called $INCOMP$. To avoid writing big arrays and a big data file, we can declare it as a dynamic array to only write where there are incompatibilities.

$$INCOMP_{jh} = \begin{cases} 1 & \text{Courses } j \text{ and } h \text{ are incompatible} \quad \forall j, h \in J \quad \text{and} \quad j \neq h \\ 0 & \text{o.w.} \end{cases}$$

Up to this point we are only avoiding to have overlaps between courses but we are missing to accomplish taking into account the structures for each course. As mentioned before, we can solve this by modifying the solution space for our variable $plan$. Still, by writing one-by-one as it is done for small instances won't work out for our problem. Therefore, if we add this as another set of constraints, lets call them "Structures" which would make possible to take out from the possible values for $plan$ that can't be assigned for each course. Lets define $STRUCT_{ji}$ as a parameter that takes a value equal to one if the course j in J cannot be assigned the schedule i in I.

$$STRUCT_{ji} = \begin{cases} 1 & \text{Course } j \text{ cannot be assigned the schedule } i \quad \forall j \in J \quad i \in I \\ 0 & \text{o.w.} \end{cases}$$

Therefore, for writing the constraints at the model, we can get the following:

- For the INCOMPATIBILITIES set of constraints:

$$plan_j \neq plan_k \quad \forall j,k \in J \mid \exists \quad INCOMP_{jk}$$

- For the Structures set of constraints:

$$plan_j \neq i \quad \forall j \in J, i \in I \mid \exists \quad STRUCT_{ji}$$

The implementation of this model will be shown on the next section.

5 Results

In this section, we present the way in which the proposals presented above are operationalized. To achieve this, simple examples are presented first as an illustration followed by the instances we studied more in detail. This section will be divided as follows:

- One-section courses Integer Programming example.
- Multiple-section courses Integer Programming example.
- Constraint Programming example.
- Instances solutions with Integer programming.
- Instances solutions with Constraint programming.

5.1 One-section courses Integer Programming examples.

Considering the integer programming model described on subsection 4.1, we coded it in XPRESS IVE. We can view the code for the model at figure 2.

```
model Courses
uses "mmxprs";
declarations
    n: integer   !Number of Blocks (|I|)
    m: integer   !Number of Courses (|J|)
    k: integer   !Number of professors
end-declarations
initializations from "DINEP2020.txt"
    n m k h
end-initializations
I:=1..n !set of blocks
J:=1..m !set of courses
declarations
    K:array(1..k) of set of integer    !professor that teaches course j
    JS:array(S) of set of integer      !courses from term s
    IJ:array(J) of set of integer      !blocks that can be assigned to course j
    x:dynamic array(range,range) of mpvar !if course j is assigned to course i
    z:linctr
end-declarations
initializations from 'DINEP2020.txt'
    K JS IJ
end-initializations
forall(j in J, i in IJ(j))create(x(i,j))

!ASSIGN TO EACH COURSE J AN TIME SLOT
forall(j in J)  sum(i in IJ(j)|exists(x(i,j))) x(i,j) = 1
!NO OVERLAPS WITH TEACHERS
forall(i in I,k0 in 1..k)sum(j in K(k0)| exists(x(i,j))) x(i,j) <= 1
!forall(i in I,k0 in 1..k)sum(j in K(k0)) x(i,j) <= 1
!NO OVERLAPS WITHIN COURSES FROM THE SAME TERM
forall(i in I, s in S)  sum(j in JS(s)|exists(x(i,j)))x(i,j) <= 1
!BINARY
forall(i in I, j in J|exists(x(i,j)))x(i,j) is_binary
```

Figure 2: Model written in Xpress IVE.

First, we explain how the construction of the data file is realized and after that, we present some examples. We worked in the creation of three parameters: K, JS and IJ. For K we have an array of size k (which remember, is the number of teachers) and for each position in K we have the set of courses that each teacher

has assigned to teach. This means that we assign for each teacher a number, say one is assigned to teacher one therefore at K in position one we find all the courses they teach, coded too with numbers. The next parameter is JS, it is for the courses at a certain term, this means we have an array of size 8, just as S but in each row of it we have the set of courses (coded by numbers) that correspond to named term. At last, IJ is an array size J, which has the possible slots in which we can schedule each course j in J. This parameter in other words can be described, for course four at row four, we find all the possible slots in which it can be programmed. We need to identify those slots by checking the course's structure and its term where they belong as sometimes this also inferes in how a course can be scheduled. We can appreciate how K and JS are represented at the data file at image 3 and part of IJ at the data file at image 4.

```
K: [
[7 20 25]          !!1
[1 16]             !!2
[2]                !!3
[3 17]             !!4
[4 5]              !!5
[6 10]             !!6
[7]                !!7
[9 14 18 19]       !!8
[11 13 22]         !!9
[12 15 23 24]      !!10
[21]               !!11
]
JS:[[]                          !!(1)
    []                          !!(2)
    [1 2]                       !!(3)
    [3 4 5 6 7]                 !!(4)
    [8 9 10 12]                 !!(5)
    [11 13 14 15 16 17]         !!(6)
    [18 23]                     !!(7)
    [19 20 21 22 24 25]         !!(8)
]
```

Figure 3: Example for how K and JS are written at the data file using data for the instance of one-section.

```
IJ: [   [1 2 3 4 5 6 7 8 9 14 15 16 17 18 19]  !!Course 1
    [147 148 149 150 151 152 75 76 77 79 80 81 83 84 87 88 89 91 92 93 95 96 99 100 101 103 104 105 107 108
    [1 2 3 4 5 6 7 8 9 14 15 16 17 18 19]  !!Course 3
    [1 2 3 4 5 6 7 8 9 14 15 16 17 18 19]  !!Course 4
    [36 37 38 39 40 41 42 43 44 49 50 51 52 53 54 55 56 62 63 64 65 66 67 68 69]  !!Course 5
    [1 2 3 4 5 6 7 8 9 14 15 16 17 18 19]  !!Course 6
    [1 2 3 4 5 6 7 8 9 14 15 16 17 18 19]  !!Course 7
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 8
    [42 43 44 45 46 47 48 56 57 58 59 60 61 69 70 71 72 73 74]  !!Course 9
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 10
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 11
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 12
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 13
    [42 43 44 45 46 47 48 56 57 58 59 60 61 69 70 71 72 73 74]  !!Course 14
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 15
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 16
    [151 152 153 154 155 81 82 85 86 89 90 93 94 97 98 101 102 105 106 109 110 113 114 117 118 121 122 125 1
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 18
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 19
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 20
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 21
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 22
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 23
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 24
    [10 11 12 13 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  !!Course 25
]
```

Figure 4: Example for IJ is written at the data file using data for the instance of one-section.

To illustrate the operation of the model, we can look at the next example, to respect confidentiality we use generic names for the courses and the teachers. Remember that we are assuming there is only need for one section of each course, this means that we don't have the full number of courses that is needed but we could see if the constraints were working on are what we need. Therefore, we have for I, 400 different slots we could use to assign courses, the number of courses to be programmed are 25, (J) and we have 11 teachers.

Our data file was created following what we have previously explained, assigning for each of the professors a list of courses he or she was going to teach in the semester, for example, having teacher one in the row one of our K we had the information he had to teach courses 7, 20 and 25, therefore at row one of the parameter we could see listed 7, 20 and 25. This was repeated for all of our 11 teachers.

Now for the two parameters related to the courses by its' term they belonged to or the structure it had, the parameters are as follows. We listed all the courses we were going to include from 1 to 25 and for each of them we identified the term it was suggested for students to enroll in it and the structure it had. Also, we wrote the teacher assigned to the course to complete the information of each course. With this list, we had easier the job of knowing the courses that belonged to each semester, for the creation of JS, and the slots that were possible for each course to be scheduled at, for the creation of IJ. Say, at JS, if courses 8, 9, 10 and 12 belonged to semester 5, at row 5 of this parameter we can find those semesters listed. As for the IJ parameter, having at our list that course 6 was a structure 1 course from semester 4, at row 6 of our parameter we could find all the number of slots to which this course could be assigned, from 1 to 9 and from 14 to 19. Paying close attention to figures 3 and 4 we can see said information.

With this done, we run our model, obtaining the x_{ij} that are equal to 1, this

means that the course j will be assigned to the block i. Having this listed we can now see if the assignation is fulfilling the constraints.

The results were written into an excel sheet, with a table to visualize the course, to which block it was assigned and other information such as term and professor assigned to it. We would write the schedule related to the number of block (I). To visualize for overlaps was hard so after having this we could build the timetables per term, to see that effectively there were no overlaps within courses from the same semester. This can be seen at figure 5. For easier visualization, we have the courses from the different semesters on different colors per semester.

Also we can see the timetables organized for each professor schedule at figure 6, here the courses have different colors depending on the professor they have assigned.

Course J	Name	Block I	Term	Professor	Schedule
1	Course 1	1	3	2	M, W, F 7:00- 7:50
2	Course 2	149	3	3	Tu, Th 10:00- 11:15
3	Course 3	2	4	4	M, W, F 8:00- 8:50
4	Course 4	3	4	5	M, W, F 9:00- 9:50
5	Course 5	66	4	5	Tu, Th 11:00-12:40
6	Course 6	4	4	6	M, W, F 10:00- 10:50
7	Course 7	1	4	1	M, W, F 7:00- 7:50
8	Course 8	12	5	7	M, W, F 18:00- 18:50
9	Course 9	43	5	8	M, Tu, W, Th 14:00- 14:50
10	Course 10	11	5	6	M, W, F 17:00- 17:50
11	Course 11	11	6	9	M, W, F 17:00- 17:50
12	Course 12	10	5	10	M, W, F 16:00- 16:50
13	Course 13	10	6	9	M, W, F 16:00- 16:50
14	Course 14	42	6	8	M, Tu, W, Th 13:00- 13:50
15	Course 15	12	6	10	M, W, F 18:00- 18:50
16	Course 16	13	6	2	M, W, F 19:00- 19:50
17	Course 17	155	6	4	Tu, Th 19:00- 20:15
18	Course 18	10	7	8	M, W, F 16:00- 16:50
19	Course 19	11	8	8	M, W, F 17:00- 17:50
20	Course 20	12	8	1	M, W, F 18:00- 18:50
21	Course 21	13	8	11	M, W, F 19:00- 19:50
22	Course 22	20	8	9	Tu, Th 16:00- 17:15
23	Course 23	11	7	10	M, W, F 17:00- 17:50
24	Course 24	21	8	10	Tu, Th 17:30- 18:45
25	Course 25	10	8	1	M, W, F 16:00- 16:50

Term 3	Monday	Tuesday	Wednesday	Thursday	Friday
07:00	Course 1		Course 1		Course 1
07:30					
08:00					
08:30					
09:00					
09:30					
10:00		Course 2		Course 2	
10:30					
11:00					
11:30					
12:00					

Term 4	Monday	Tuesday	Wednesday	Thursday	Friday
07:00	Course 7		Course 7		Course 7
07:30					
08:00	Course 3		Course 3		Course 3
08:30					
09:00	Course 4		Course 4		Course 4
09:30					
10:00	Course 6		Course 6		Course 6
10:30					
11:00		Course 5		Course 5	
11:30					
12:00					
12:30					
13:00					
13:30					

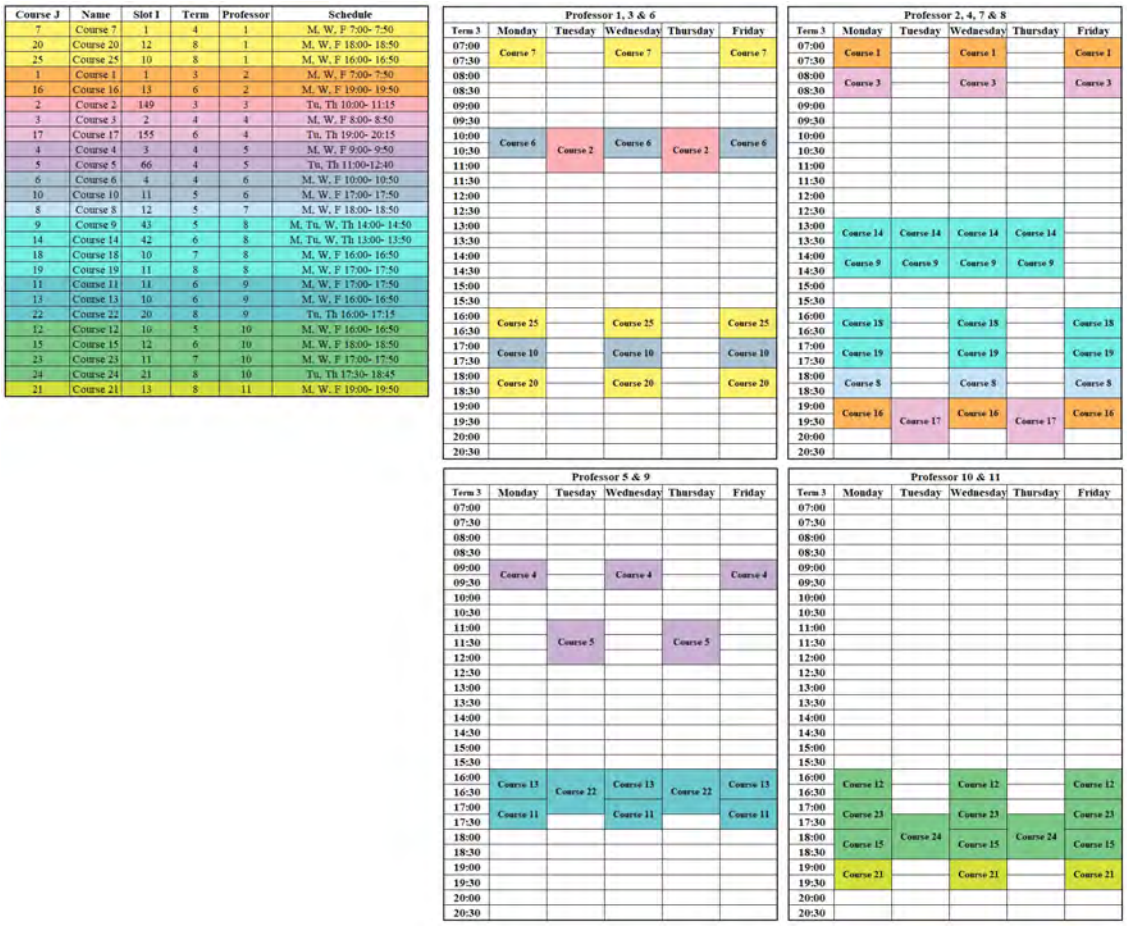
Term 5	Monday	Tuesday	Wednesday	Thursday	Friday
13:00					
13:30					
14:00	Course 9	Course 9	Course 9	Course 9	
14:30					
15:00					
15:30					
16:00	Course 12		Course 12		Course 12
16:30					
17:00	Course 10		Course 10		Course 10
17:30					
18:00	Course 8		Course 8		Course 8
18:30					
19:00					
19:30					

Term 6	Monday	Tuesday	Wednesday	Thursday	Friday
12:00					
12:30					
13:00	Course 14	Course 14	Course 14	Course 14	
13:30					
14:00					
14:30					
15:00					
15:30					
16:00	Course 13		Course 13		Course 13
16:30					
17:00	Course 11		Course 11		Course 11
17:30					
18:00	Course 15		Course 15		Course 15
18:30					
19:00	Course 16	Course 17	Course 16	Course 17	Course 16
19:30					
20:00					
20:30					

Term 7	Monday	Tuesday	Wednesday	Thursday	Friday
15:00					
15:30					
16:00	Course 18		Course 18		Course 18
16:30					
17:00	Course 23		Course 23		Course 23
17:30					
18:00					
18:30					

Term 8	Monday	Tuesday	Wednesday	Thursday	Friday
15:00					
15:30					
16:00	Course 25	Course 22	Course 25	Course 22	Course 25
16:30					
17:00	Course 19		Course 19		Course 19
17:30					
18:00	Course 20	Course 24	Course 20	Course 24	Course 20
18:30					
19:00	Course 21		Course 21		Course 21
19:30					
20:00					
20:30					

Figure 5: Summary of the results for the example considering one section, timetables organized by term.



Course J	Name	Slot I	Term	Professor	Schedule
7	Course 7	1	4	1	M, W, F 7:00- 7:50
20	Course 20	12	8	1	M, W, F 18:00- 18:50
25	Course 25	10	8	1	M, W, F 16:00- 16:50
1	Course 1	1	3	2	M, W, F 7:00- 7:50
16	Course 16	13	6	2	M, W, F 19:00- 19:50
2	Course 2	149	3	3	Tu, Th 10:00- 11:15
3	Course 3	2	4	4	M, W, F 8:00- 8:50
17	Course 17	155	6	4	Tu, Th 19:00- 20:15
4	Course 4	3	4	5	M, W, F 9:00- 9:50
5	Course 5	66	4	5	Tu, Th 11:00-12:40
6	Course 6	4	4	6	M, W, F 10:00- 10:50
10	Course 10	11	5	6	M, W, F 17:00- 17:50
8	Course 8	12	5	7	M, W, F 18:00- 18:50
9	Course 9	43	5	8	M, Tu, W, Th 14:00- 14:50
14	Course 14	42	6	8	M, Tu, W, Th 13:00- 13:50
18	Course 18	10	7	8	M, W, F 16:00- 16:50
19	Course 19	11	8	8	M, W, F 17:00- 17:50
11	Course 11	11	6	9	M, W, F 17:00- 17:50
13	Course 13	10	6	9	M, W, F 16:00- 16:50
22	Course 22	20	8	9	Tu, Th 16:00- 17:15
12	Course 12	10	5	10	M, W, F 16:00- 16:50
15	Course 15	12	6	10	M, W, F 18:00- 18:50
23	Course 23	11	7	10	M, W, F 17:00- 17:50
24	Course 24	21	8	10	Tu, Th 17:30- 18:45
21	Course 21	13	8	11	M, W, F 19:00- 19:50

Figure 6: Summary of the results for the example considering one section, timetables organized by professor.

As we can observe at figures 5 and 6, we can be sure that the model is assigning the courses to a schedule that respects the rules and constraints previously discussed. This can be checked, for example, if we look into the classes for the 4th term of Career 2 (figure 5) we can see that students at their 4th term can take all their courses as there is no overlaps and the latest class they have is Tuesday and Thursday from 11:00 to 12:40 meaning we are funneling that semester in that specific schedule. Also, we can check an example for a professor's schedule (figure 6), we can look at professor 5, at the table on figure 6 with the courses in purple color, and at the timetable with the title of "Professor 5 & 9" we can see her/his two courses are far from being overlapped.

This was repeated with the other two careers from the department in study to ensure that our model works with different sizes of data files. Don't forget here, we are testing with the assumption of single section courses. The next career to be tested was the smallest one perceived of the department in number of students (then the assumption of only one section per course wouldn't make any problem),

this one we will call Career 1, with information of what we could observe from Spring 2020 also, for simplifying it further we only did it with the information of courses from one study plan which represented mostly courses from term 4 to 8. Due confidentiality reasons all information of course's name, teacher's name will be with generic names.

Again, all the process to create the data files was repeated and the model ran in Xpress IVE, from which we obtain which course $j \in J$ is assigned to which block $i \in I$. Then we moved the data to Excel to obtain the table with the different time slots assigned, as can be seen at figure 7. There we can also see all the courses arranged in timetables divided by term to appreciate the final schedules in a much clearer way, at this figure we can also see the information for each course.

Course J	Name	Block I	Term	Professor	Schedule
1	Course 1	17	4	6	Tu, Th 11:30- 12:45
2	Course 2	10	5	1	M,W, F 16-16:50
3	Course 3	10	6	2	M,W, F 16-16:50
4	Course 4	74	6	3	Tu, Th 19:00- 20:40
5	Course 5	11	7	4	M,W, F 17-17:50
6	Course 6	10	8	4	M,W, F 16-16:50
7	Course 7	20	8	2	Tu, Th 16:00- 17:15
8	Course 8	13	8	1	M,W, F 19-19:50
9	Course 9	12	8	5	M,W, F 18-18:50
10	Course 10	11	8	3	M,W, F 17-17:50
11	Course 11	21	8	7	Tu, Th 17:30- 18:45

Term 4	Monday	Tuesday	Wednesday	Thursday	Friday
10:00					
10:30					
11:00					
11:30		Course 1		Course 1	
12:00					
12:30					
13:00					
13:30					

Term 5	Monday	Tuesday	Wednesday	Thursday	Friday
14:00					
14:30					
15:00					
15:30					
16:00	Course 2		Course 2		Course 2
16:30					
17:00					
17:30					

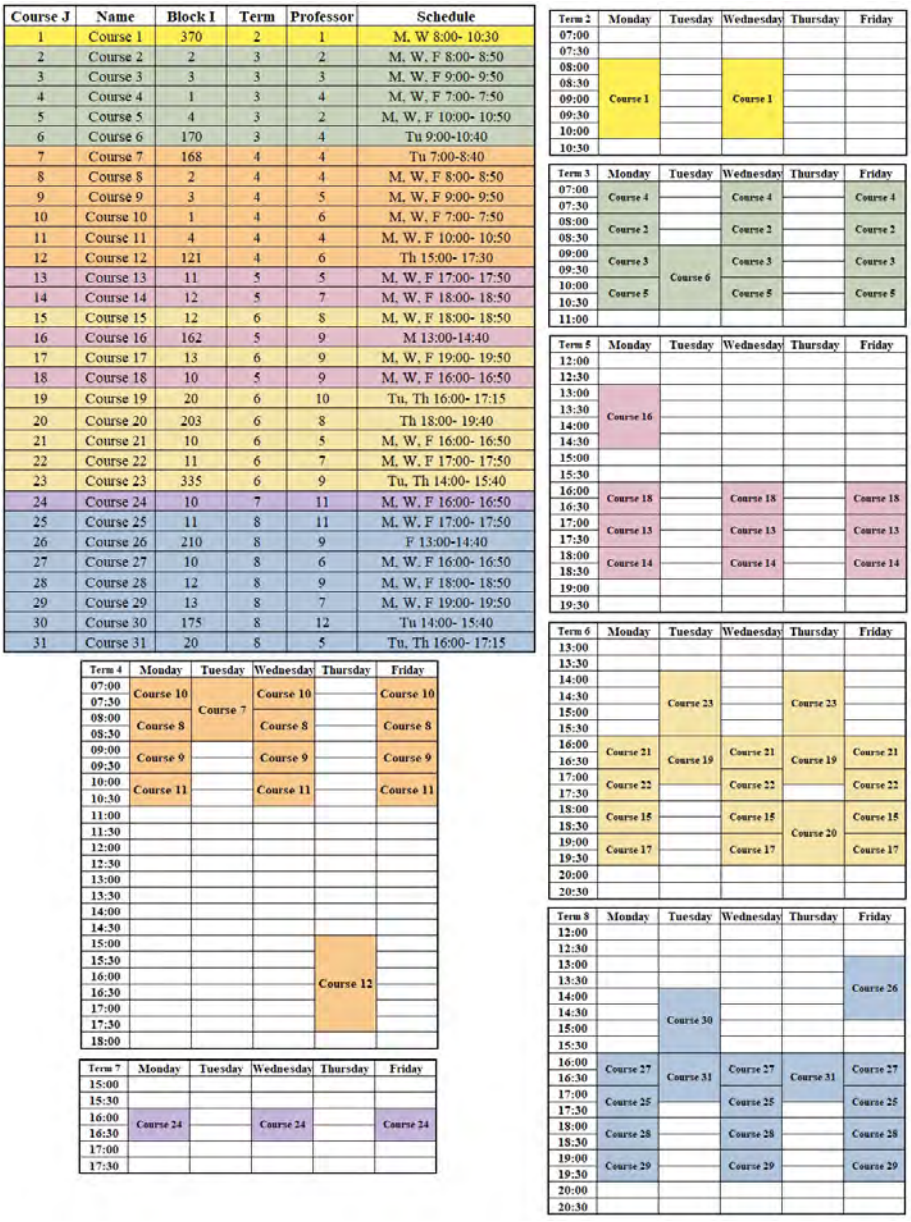
Term 6	Monday	Tuesday	Wednesday	Thursday	Friday
13:00					
13:30					
14:00					
14:30					
15:00					
15:30					
16:00	Course 3		Course 3		Course 3
16:30					
17:00					
17:30					
18:00					
18:30					
19:00		Course 4		Course 4	
19:30					
20:00					
20:30					

Term 7	Monday	Tuesday	Wednesday	Thursday	Friday
14:00					
14:30					
15:00					
15:30					
16:00					
16:30					
17:00	Course 5		Course 5		Course 5
17:30					
18:00					
18:30					
19:00					
19:30					
20:00					
20:30					

Term 8	Monday	Tuesday	Wednesday	Thursday	Friday
14:00					
14:30					
15:00					
15:30					
16:00	Course 6	Course 7	Course 6	Course 7	Course 6
16:30					
17:00	Course 10		Course 10		Course 10
17:30		Course 11		Course 11	
18:00	Course 9		Course 9		Course 9
18:30					
19:00	Course 8		Course 8		Course 8
19:30					
20:00					
20:30					

Figure 7: Summary of results for one section instance of Career 1 with data corresponding to Spring 2020.

The process was repeated for a third career. The results obtained for Career 3 are at figure 8. We decided to separate the careers first to see how our model behaves with different sizes of data files, for career 1 we can see we only need 11 courses to be assigned, while for career 2 we had 25, and for career 3 we had 31 courses. The computer times for the model to find a feasible solution are very small, making it fast to obtain our results in a list but the tough part is in making the timetables for them to be easy to check for any possible overlap which wasn't found.



Course J	Name	Block I	Term	Professor	Schedule
1	Course 1	370	2	1	M, W 8:00- 10:30
2	Course 2	2	3	2	M, W, F 8:00- 8:50
3	Course 3	3	3	3	M, W, F 9:00- 9:50
4	Course 4	1	3	4	M, W, F 7:00- 7:50
5	Course 5	4	3	2	M, W, F 10:00- 10:50
6	Course 6	170	3	4	Tu 9:00-10:40
7	Course 7	168	4	4	Tu 7:00-8:40
8	Course 8	2	4	4	M, W, F 8:00- 8:50
9	Course 9	3	4	5	M, W, F 9:00- 9:50
10	Course 10	1	4	6	M, W, F 7:00- 7:50
11	Course 11	4	4	4	M, W, F 10:00- 10:50
12	Course 12	121	4	6	Th 15:00- 17:30
13	Course 13	11	5	5	M, W, F 17:00- 17:50
14	Course 14	12	5	7	M, W, F 18:00- 18:50
15	Course 15	12	6	8	M, W, F 18:00- 18:50
16	Course 16	162	5	9	M 13:00-14:40
17	Course 17	13	6	9	M, W, F 19:00- 19:50
18	Course 18	10	5	9	M, W, F 16:00- 16:50
19	Course 19	20	6	10	Tu, Th 16:00- 17:15
20	Course 20	203	6	8	Th 18:00- 19:40
21	Course 21	10	6	5	M, W, F 16:00- 16:50
22	Course 22	11	6	7	M, W, F 17:00- 17:50
23	Course 23	335	6	9	Tu, Th 14:00- 15:40
24	Course 24	10	7	11	M, W, F 16:00- 16:50
25	Course 25	11	8	11	M, W, F 17:00- 17:50
26	Course 26	210	8	9	F 13:00-14:40
27	Course 27	10	8	6	M, W, F 16:00- 16:50
28	Course 28	12	8	9	M, W, F 18:00- 18:50
29	Course 29	13	8	7	M, W, F 19:00- 19:50
30	Course 30	175	8	12	Tu 14:00- 15:40
31	Course 31	20	8	5	Tu, Th 16:00- 17:15

Figure 8: Summary of results for one section instance of Career 3 with data corresponding to Spring 2020.

Once we checked each career for separate and confirming that the model is working correctly, we moved onto the next step which was to find out how to manage a multiple-section instance, this is shown in the next subsection.

5.2 Multiple-section courses Integer Programming example.

For solving the Multiple- Section Courses instance, we decided to add at the number of courses the extra sections or groups, as if they were independent courses. So, the idea was that the size of J was going to grow, in other words, we would have the list of all the courses considering every section, so that if for course 3, at row 3 when having only one section, we needed 2 sections at row 3 we would have course 3-1 and at row 4 course 3-2, each with their own data and row at the parameters. Each section would receive a block to be assigned with.

Adding the number of sections to the list of courses did that all sections for a course shouldn't be scheduled at the same block as we consider that a course will always be taught by the same teacher, this way we wouldn't have problems because there wouldn't be overlaps nor in courses of the same term nor because of the teacher. Almost right, as in some cases, the same course was taught by the same teacher at all sections but, in other cases, we had for the same course, different teachers giving the course. Therefore, at those cases when looking at the proposed schedules for the students there might appear big spaces because in between the courses we would have the different schedules for a course assigned and students only needed to be at one.

For solving this situation, we decided to change how JS worked. Instead of it showing the courses that belonged for each semester, we would make it of size h where h stands for the number of different array of classes that might be needed. As it is shown in figure 9 we have an example of how we write parameter JS at the data file. We can see we have a value of 11 for h as JS has 11 rows. Also, we can see how this new way to use the parameter is used. If we look at rows 2, 3 and 4 we can see as comments that they are denoted as 4.1, 4.2 and 4.3 respectively, and for the values inside the sets for each row we can see for example that we have for row 2 an 8, for row 3 a 9 and for row 4 a 10, this means that 8,9 and 10 stand for the same course but in different sections.

```
JS:[[1 2]                          !!3
    [3 5 7 8 11]                   !!4.1
    [4 6 7 9 11]                   !!4.2
    [3 6 7 10 11]                  !!4.3
    [12 13 14 17]                  !!5
    [15 18 20 22 25]               !!6.1
    [16 19 21 23 26]               !!6.2
    [15 19 20 24 27]               !!6.3
    [28 34 35]                     !!7
    [29 30 31 32 34 35 36]         !!8.1
    [29 30 31 33 34 35 36]         !!8.2
]
```

Figure 9: Parameter JS at data file with information of semesters with courses with multiple-sections.

Also, this rows correspond to the fourth term of career two and we number them to indicate that we can have 3 versions of schedule for that term.

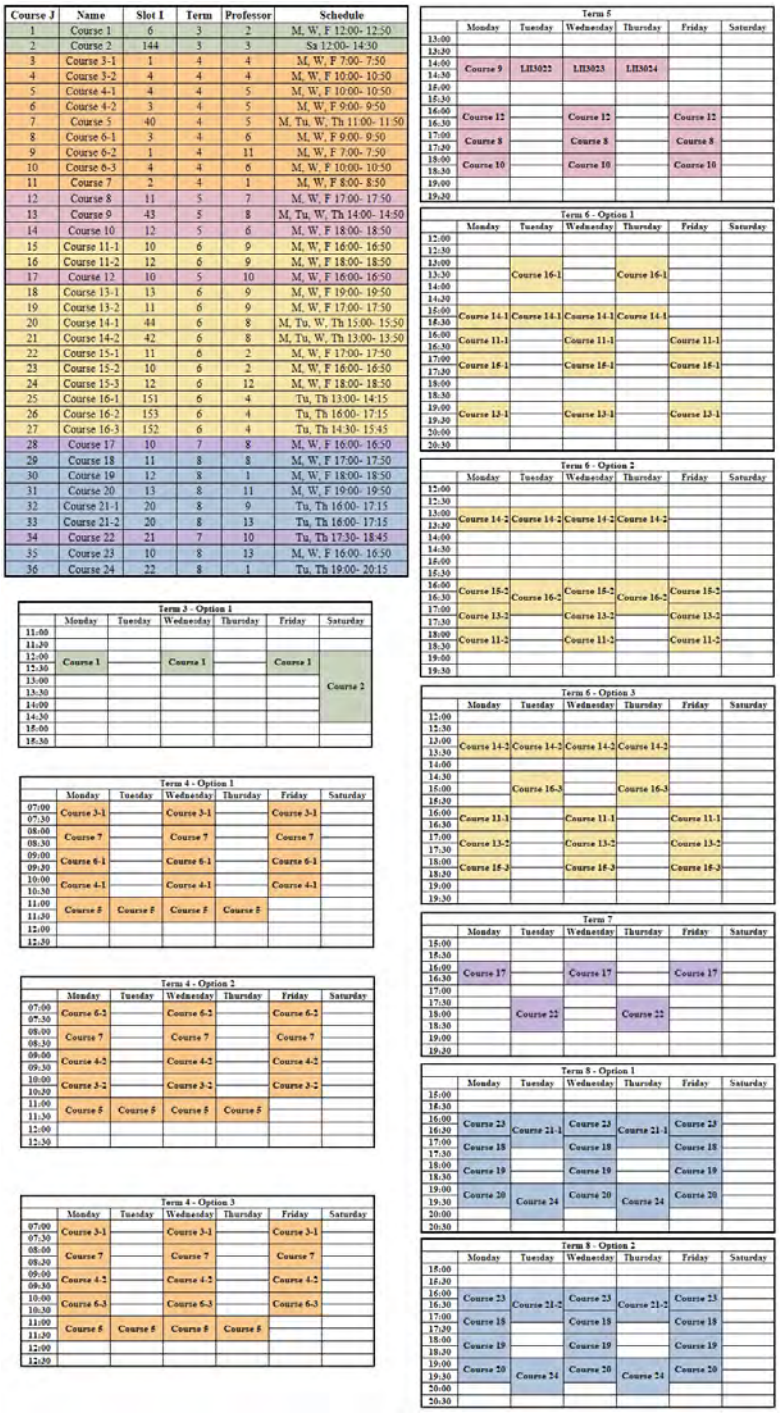
The creation of the other two parameters ($KandIJ$) was the same as with the one-section example. Now, to show how the Multiple-Section Courses program works, considering only one career (career 2). We have once again had to do the list for our professors and courses to be included as in the previous subsection. Once again, this was done to keep track and order of what was to be involved in the model and avoid making mistakes of assignation.

Notice that we are using the same career as an example as the first one shown in the previous subsection, the difference is the number of sections used, previously we did it for only one section per course and now we have the real number of sections that were offered at the term of Spring 2020, again, for confidentiality we will use generic names for the courses. The difference can be seen at the size of J, before we had J of size 25 as we had 25 courses, now we face a J of size 36 as the new sections were added.

Having all ready, we ran the model obtaining the values for x_{ij} which were processed in excel to obtain what we can see in figure 10. We can see the different possibilities for each term to have its courses programmed, the colors are for easily identifying the term a course belongs to. Once again we can see the model did its work at fulfilling the rules for class-hours depending on the term and the structures.

We can see the case of having two sections with different teachers assigned at the same schedule when looking at the timetables for "Term 8- Option 1" and "Term 8- Option 2" everything looks the same because we only have one double section course which is Course 21, section one is taught by professor 9 and section two by professor 13 allowing the model to assign them to the same schedule and end up with similar schedules for the students either they choose section 1 or 2 at this course.

This was the only example done for the multiple-section solution with integer programming. Later will be shown the instances studied which are involving multiple sections and different careers together in a single data file.



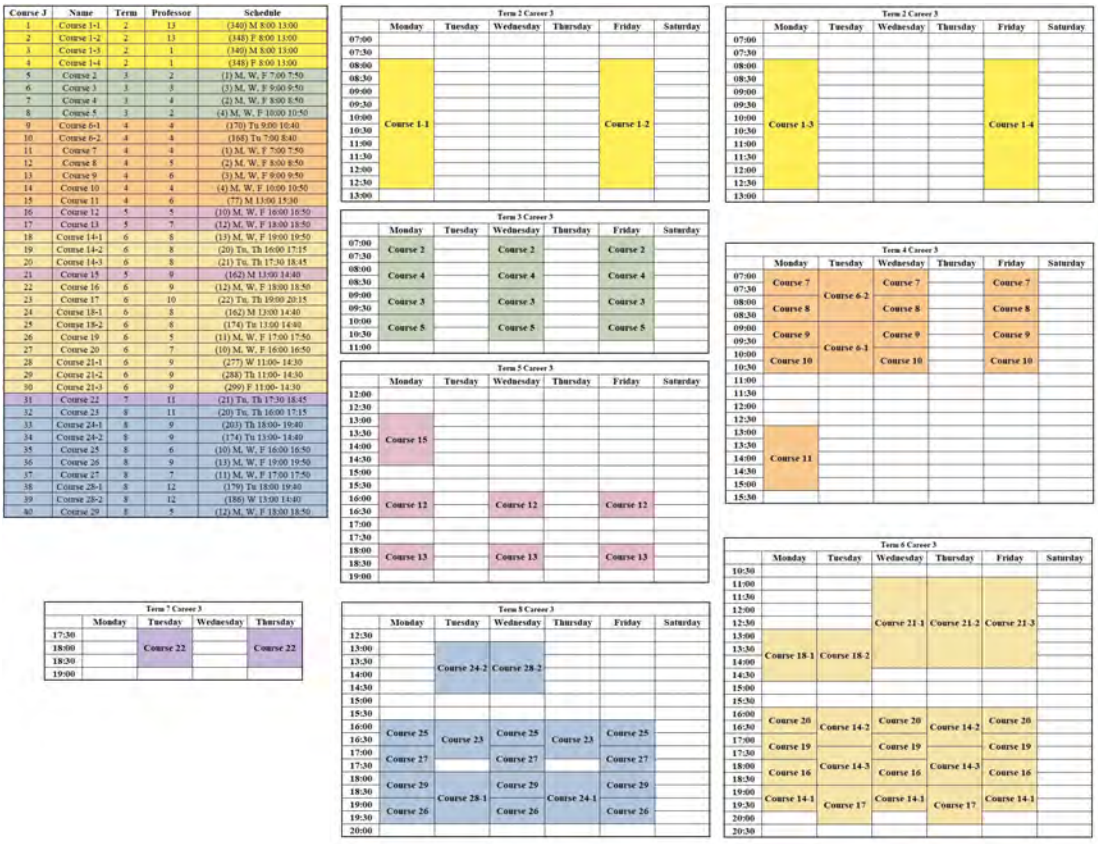
Course J	Name	Slot I	Term	Professor	Schedule
1	Course 1	6	3	2	M, W, F 12:00- 12:50
2	Course 2	144	3	3	Sa 12:00- 14:30
3	Course 3-1	1	4	4	M, W, F 7:00- 7:50
4	Course 3-2	4	4	4	M, W, F 10:00- 10:50
5	Course 4-1	4	4	5	M, W, F 10:00- 10:50
6	Course 4-2	3	4	5	M, W, F 9:00- 9:50
7	Course 5	40	4	5	M, Tu, W, Th 11:00- 11:50
8	Course 6-1	3	4	6	M, W, F 9:00- 9:50
9	Course 6-2	1	4	11	M, W, F 7:00- 7:50
10	Course 6-3	4	4	6	M, W, F 10:00- 10:50
11	Course 7	2	4	1	M, W, F 8:00- 8:50
12	Course 8	11	5	7	M, W, F 17:00- 17:50
13	Course 9	43	5	8	M, Tu, W, Th 14:00- 14:50
14	Course 10	12	5	6	M, W, F 18:00- 18:50
15	Course 11-1	10	6	9	M, W, F 16:00- 16:50
16	Course 11-2	12	6	9	M, W, F 18:00- 18:50
17	Course 12	10	5	10	M, W, F 16:00- 16:50
18	Course 13-1	13	6	9	M, W, F 19:00- 19:50
19	Course 13-2	11	6	9	M, W, F 17:00- 17:50
20	Course 14-1	44	6	8	M, Tu, W, Th 15:00- 15:50
21	Course 14-2	42	6	8	M, Tu, W, Th 13:00- 13:50
22	Course 15-1	11	6	2	M, W, F 17:00- 17:50
23	Course 15-2	10	6	2	M, W, F 16:00- 16:50
24	Course 15-3	12	6	12	M, W, F 18:00- 18:50
25	Course 16-1	151	6	4	Tu, Th 13:00- 14:15
26	Course 16-2	153	6	4	Tu, Th 16:00- 17:15
27	Course 16-3	152	6	4	Tu, Th 14:30- 15:45
28	Course 17	10	7	8	M, W, F 16:00- 16:50
29	Course 18	11	8	8	M, W, F 17:00- 17:50
30	Course 19	12	8	1	M, W, F 18:00- 18:50
31	Course 20	13	8	11	M, W, F 19:00- 19:50
32	Course 21-1	20	8	9	Tu, Th 16:00- 17:15
33	Course 21-2	20	8	13	Tu, Th 16:00- 17:15
34	Course 22	21	7	10	Tu, Th 17:30- 18:45
35	Course 23	10	8	13	M, W, F 16:00- 16:50
36	Course 24	22	8	1	Tu, Th 19:00- 20:15

Figure 10: Schedules obtained for career two with multiple sections.

5.3 Constraint Programming example.

As it was made for the integer programming approach, we tested too how the constraint programming model worked with a small instance. This was to ensure everything was fulfilled in a smaller instance and to correct if it was needed.

For this, we ran the model with information of Career 3 from the term of Spring considering all the groups needed for each of the courses. Our model didn't need a difference for running with an assumption of only single-section courses or multiple-sections. The data file was created and the model ran. The summary of what was obtained as well as the timetables created for easy visualization can be seen at figure 11.



Course J	Name	Term	Professor	Schedule
1	Course 1-1	2	13	(340) M 8:00 13:00
2	Course 1-2	2	13	(348) F 8:00 13:00
3	Course 1-3	2	1	(349) M 8:00 13:00
4	Course 1-4	2	1	(348) F 8:00 13:00
5	Course 2	3	2	(1) M, W, F 7:00 7:50
6	Course 3	3	3	(3) M, W, F 9:00 9:50
7	Course 4	3	4	(2) M, W, F 8:00 8:50
8	Course 5	3	2	(4) M, W, F 10:00 10:50
9	Course 6-1	4	4	(170) Tu 9:00 10:40
10	Course 6-2	4	4	(168) Tu 7:00 8:40
11	Course 7	4	4	(1) M, W, F 7:00 7:50
12	Course 8	4	5	(2) M, W, F 8:00 8:50
13	Course 9	4	6	(3) M, W, F 9:00 9:50
14	Course 10	4	4	(4) M, W, F 10:00 10:50
15	Course 11	4	6	(77) M 13:00 15:30
16	Course 12	5	5	(10) M, W, F 16:00 16:50
17	Course 13	5	7	(12) M, W, F 18:00 18:50
18	Course 14-1	6	8	(13) M, W, F 19:00 19:50
19	Course 14-2	6	8	(20) Tu, Th 16:00 17:15
20	Course 14-3	6	8	(21) Tu, Th 17:30 18:45
21	Course 15	5	9	(162) M 13:00 14:40
22	Course 16	6	9	(12) M, W, F 18:00 18:50
23	Course 17	6	10	(22) Tu, Th 19:00 20:15
24	Course 18-1	6	8	(162) M 13:00 14:40
25	Course 18-2	6	8	(174) Tu 13:00 14:40
26	Course 19	6	5	(11) M, W, F 17:00 17:50
27	Course 20	6	7	(10) M, W, F 16:00 16:50
28	Course 21-1	6	9	(277) W 11:00- 14:30
29	Course 21-2	6	9	(288) Th 11:00- 14:30
30	Course 21-3	6	9	(299) F 11:00- 14:30
31	Course 22	7	11	(21) Tu, Th 17:30 18:45
32	Course 23	8	11	(20) Tu, Th 16:00 17:15
33	Course 24-1	8	9	(203) Th 18:00- 19:40
34	Course 24-2	8	9	(174) Tu 13:00- 14:40
35	Course 25	8	6	(10) M, W, F 16:00 16:50
36	Course 26	8	9	(13) M, W, F 19:00 19:50
37	Course 27	8	7	(11) M, W, F 17:00 17:50
38	Course 28-1	8	12	(179) Tu 18:00 19:40
39	Course 28-2	8	12	(186) W 13:00 14:40
40	Course 29	8	5	(12) M, W, F 18:00 18:50

Figure 11: Summary of the results obtained for Career 3. Considering all the sections needed at Spring 2020.

We can view that all the courses could be right scheduled, no overlaps over the courses from same semester and all the courses were assigned too. Thus we proceeded to make the data files for the whole instances we wanted to study, this will be discussed later.

5.4 Instances solutions with Integer programming.

Once we are sure that our models work properly, due the previous examples done to ensure the right performance, we proceed to test them with real instances. Hence, we worked with our multiple sections instance with the data files prepared to later compare it with what courses were offered at a certain semester in the university, once again, all the names for the courses and teachers are confidential and consequently have generic names. We decided to run the whole set, with information of the 3 careers managed by the department, two times, first with the published information for Spring 2020, and later with information of Fall 2020.

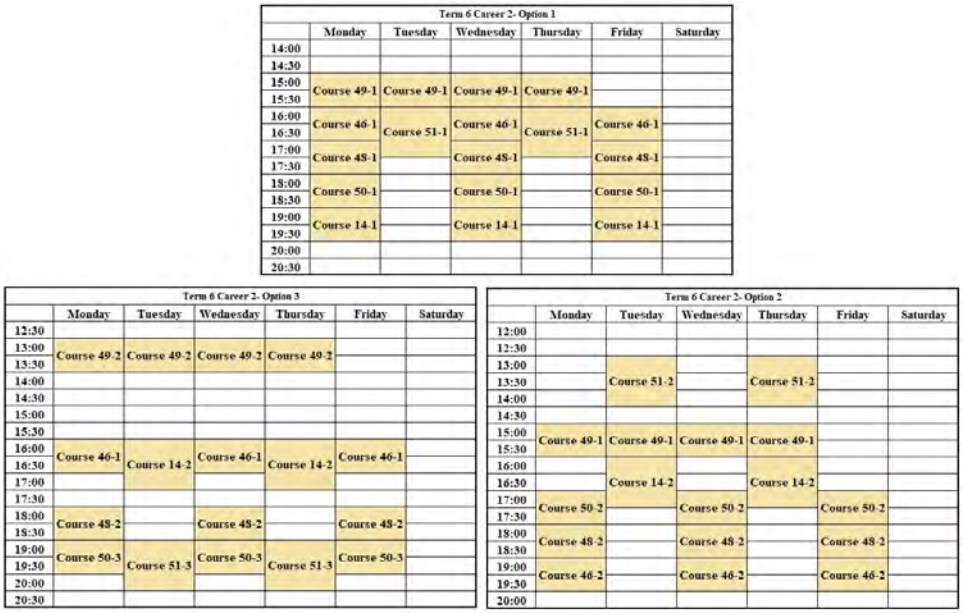
For the data related to Spring 2020, we have 400 possible time-slots in which we can schedule classes, the size of J is 80, having 57 courses. We can see at figure 12 the summary of the results obtained, followed by a short sample of the different schedules obtained and afterwards is the review of the timetables obtained divided by semesters and careers, some of them grouped for simplification.

Course J	Name	Slot I	Term	Professor	Schedule
1	Course 1-1	348	2	13	F 8:00- 13:00
2	Course 1-2	340	2	13	M 8:00- 13:00
3	Course 1-3	348	2	1	F 8:00- 13:00
4	Course 1-4	340	2	1	M 8:00- 13:00
5	Course 2	1	3	2	M, W, F 7:00- 7:50
6	Course 3	2	3	3	M, W, F 8:00- 8:50
7	Course 4	3	3	4	M, W, F 9:00- 9:50
8	Course 5	4	3	2	M, W, F 10:00- 10:50
9	Course 6-1	170	4	4	Tu 9:00- 10:40
10	Course 6-2	174	4	4	Tu 13:00- 14:40
11	Course 7	1	4	4	M, W, F 7:00- 7:50
12	Course 8	3	4	5	M, W, F 9:00- 9:50
13	Course 9	4	4	6	M, W, F 10:00- 10:50
14	Course 10	2	4	4	M, W, F 8:00- 8:50
15	Course 11	77	4	6	M 13:00- 15:30
16	Course 12	12	5	5	M, W, F 18:00- 18:50
17	Course 13	11	5	7	M, W, F 17:00- 17:50
18	Course 14-1	13	6	8	M, W, F 19:00- 19:50
19	Course 14-2	20	6	8	Tu, Th 16:00- 17:15
20	Course 14-3	12	6	8	M, W, F 18:00- 18:50
21	Course 15	162	5	9	M 13:00- 14:40
22	Course 16	11	6	9	M, W, F 17:00- 17:50
23	Course 17	21	6	10	Tu, Th 17:30- 18:45
24	Course 18-1	174	6	8	Tu 13:00- 14:40
25	Course 18-2	162	6	8	M 13:00- 14:40
26	Course 19	22	6	5	Tu, Th 19:00- 20:15
27	Course 20	10	6	7	M, W, F 16:00- 16:50
28	Course 21-1	333	6	9	Tu, Th 12:00- 13:40
29	Course 21-2	267	6	9	Tu 12:00- 15:30
30	Course 21-3	331	6	9	Tu, Th 10:00- 11:40
31	Course 22	12	7	11	M, W, F 18:00- 18:50
32	Course 23	11	8	11	M, W, F 17:00- 17:50
33	Course 24-1	210	8	9	F 13:00- 14:40
34	Course 24-2	186	8	9	W 13:00- 14:40
35	Course 25	12	8	6	M, W, F 18:00- 18:50
36	Course 26	13	8	9	M, W, F 19:00- 19:50
37	Course 27	20	8	7	Tu, Th 16:00- 17:15
38	Course 28-1	186	8	12	W 13:00- 14:40
39	Course 28-2	179	8	12	Tu 18:00- 19:40
40	Course 29	10	8	5	M, W, F 16:00- 16:50

Course J	Name	Slot I	Term	Professor	Schedule
41	Course 30	2	4	19	M, W, F 8:00- 8:50
42	Course 31	10	5	14	M, W, F 16:00- 16:50
43	Course 32	10	6	15	M, W, F 16:00- 16:50
44	Course 33	12	8	15	M, W, F 18:00- 18:50
45	Course 34	13	8	14	M, W, F 19:00- 19:50
46	Course 35	20	8	17	Tu, Th 16:00- 17:15
47	Course 36	1	3	20	M, W, F 7:00- 7:50
48	Course 37	136	3	21	Sa 10:00- 12:30
49	Course 38-1	1	4	22	M, W, F 7:00- 7:50
50	Course 38-2	4	4	22	M, W, F 10:00- 10:50
51	Course 39-1	4	4	23	M, W, F 10:00- 10:50
52	Course 39-2	3	4	23	M, W, F 9:00- 9:50
53	Course 40	69	4	23	Tu, Th 14:00- 15:40
54	Course 41-1	3	4	24	M, W, F 9:00- 9:50
55	Course 41-2	1	4	15	M, W, F 7:00- 7:50
56	Course 41-3	4	4	24	M, W, F 10:00- 10:50
57	Course 42	2	4	18	M, W, F 8:00- 8:50
58	Course 43	11	5	14	M, W, F 17:00- 17:50
59	Course 44	43	5	11	M, Tu, W, Th 14:00- 14:50
60	Course 45	12	5	24	M, W, F 18:00- 18:50
61	Course 46-1	10	6	25	M, W, F 16:00- 16:50
62	Course 46-2	13	6	25	M, W, F 19:00- 19:50
63	Course 47	13	5	16	M, W, F 19:00- 19:50
64	Course 48-1	11	6	25	M, W, F 17:00- 17:50
65	Course 48-2	12	6	25	M, W, F 18:00- 18:50
66	Course 49-1	44	6	11	M, Tu, W, Th 15:00- 15:50
67	Course 49-2	42	6	11	M, Tu, W, Th 13:00- 13:50
68	Course 50-1	12	6	20	M, W, F 18:00- 18:50
69	Course 50-2	11	6	20	M, W, F 17:00- 17:50
70	Course 50-3	13	6	26	M, W, F 19:00- 19:50
71	Course 51-1	153	6	22	Tu, Th 16:00- 17:15
72	Course 51-2	151	6	22	Tu, Th 13:00- 14:15
73	Course 51-3	155	6	22	Tu, Th 19:00- 20:15
74	Course 52	12	8	18	M, W, F 18:00- 18:50
75	Course 53	13	8	15	M, W, F 19:00- 19:50
76	Course 54-1	20	8	25	Tu, Th 16:00- 17:15
77	Course 54-2	20	8	1	Tu, Th 16:00- 17:15
78	Course 55	10	7	16	M, W, F 16:00- 16:50
79	Course 56	21	8	1	Tu, Th 17:30- 18:45
80	Course 57	22	8	18	Tu, Th 19:00- 20:15

Figure 12: Summary of the results for the Multiple section instance for Careers 1,2 and 3. Considering all the sections needed at Spring 2020.

We can see there are many more timetables, this is due the high number of possibilities to take courses, we can see for example that for the sixth term of Career 2 (at figure 13, we can have 3 options of schedule, with different combinations of sections and having all the courses that are meant to be at the term.



Term 6 Career 2- Option 1

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
14:00						
14:30						
15:00	Course 49-1	Course 49-1	Course 49-1	Course 49-1		
15:30						
16:00	Course 46-1	Course 51-1	Course 46-1	Course 51-1	Course 46-1	
16:30						
17:00	Course 48-1		Course 48-1		Course 48-1	
17:30						
18:00	Course 50-1		Course 50-1		Course 50-1	
18:30						
19:00	Course 14-1		Course 14-1		Course 14-1	
19:30						
20:00						
20:30						

Term 6 Career 2- Option 3

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
12:30						
13:00	Course 49-2	Course 49-2	Course 49-2	Course 49-2		
13:30						
14:00						
14:30						
15:00						
15:30						
16:00	Course 46-1	Course 14-2	Course 46-1	Course 14-2	Course 46-1	
16:30						
17:00						
17:30						
18:00	Course 48-2		Course 48-2		Course 48-2	
18:30						
19:00	Course 50-3	Course 51-3	Course 50-3	Course 51-3	Course 50-3	
19:30						
20:00						
20:30						

Term 6 Career 2- Option 2

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
12:00						
12:30						
13:00						
13:30		Course 51-2		Course 51-2		
14:00						
14:30						
15:00	Course 49-1	Course 49-1	Course 49-1	Course 49-1		
15:30						
16:00						
16:30		Course 14-2		Course 14-2		
17:00	Course 50-2		Course 50-2		Course 50-2	
17:30						
18:00	Course 48-2		Course 48-2		Course 48-2	
18:30						
19:00	Course 46-2		Course 46-2		Course 46-2	
19:30						
20:00						

Figure 13: Options for schedule for sixth term of Career 2, programmed with integer programming.

After obtaining this, we wanted to observe if our courses were being programmed in an attractive way, that means, with less holes between classes and availability to enroll in all the courses needed per semester.

Course J	Name	Term	Professor	Schedule
1	Course 1-1	2	13	M, W, F 9:00- 10:40
2	Course 1-2	2	13	M, W, F 11:00- 12:40
3	Course 1-3	2	1	M, W, F 7:00- 8:40
4	Course 1-4	2	1	Tu, Th 17:30- 20:00
5	Course 2	3	2	Tu, Th 17:30- 18:45
6	Course 3	3	3	Tu, Th 14:30- 15:45
7	Course 4	3	4	Tu, Th 8:30- 9:45
8	Course 5	3	2	Tu, Th 17:30- 18:45
9	Course 6-1	4	4	Tu 16:00- 17:40
10	Course 6-2	4	4	Th 16:00- 17:40
11	Course 7	4	4	Tu, Th 8:30- 9:45
12	Course 8	4	5	Tu, Th 10:00- 11:15
13	Course 9	4	6	Tu, Th 11:30- 12:45
14	Course 10	4	4	Tu, Th 13:00- 14:15
15	Course 11	4	6	Tu, Th 13:00- 14:15
16	Course 12	5	5	Tu, Th 10:00- 11:15
17	Course 13	5	7	M, W, F 15:00- 15:50
18	Course 14-1	6	8	Tu, Th 10:00- 11:15
19	Course 14-2	6	8	Tu, Th 13:00- 14:15
20	Course 14-3	6	8	Tu, Th 16:00- 17:15
21	Course 15	5	9	M, W 17:00- 17:50
22	Course 16	6	9	M, W, F 16:00- 16:50
23	Course 17	6	10	Tu, Th 16:00- 17:15
24	Course 18-1	6	8	M 9:00- 10:40
25	Course 18-2	6	8	W 9:00- 10:40
26	Course 19	6	5	M, W, F 13:00- 13:50
27	Course 20	6	7	M, W, F 15:00- 15:50
28	Course 21-1	6	9	Tu, Th 8:00- 9:40
29	Course 21-2	6	9	Tu, Th 14:00- 15:40
30	Course 21-3	6	9	M, W 17:00- 18:40
31	Course 22	7	11	F 18:00- 20:30
32	Course 23	8	11	W 19:00-21:30
33	Course 24-1	8	9	Tu 16:00- 17:40
34	Course 24-2	8	9	Th 16:00- 17:40
35	Course 25	8	6	Tu 17:30- 20:00
36	Course 26	8	9	Tu, Th 7:00- 8:15
37	Course 27	8	7	Tu, Th 14:30- 15:45
38	Course 28-1	8	12	Th 16:00- 17:40
39	Course 28-2	8	12	F 16:00- 17:40
40	Course 29	8	5	Th 17:30- 20:00

Course J	Name	Term	Professor	Schedule
41	Course 30	4	19	Tu, Th 7:00- 8:15
42	Course 31	5	14	Tu, Th 11:30- 12:45
43	Course 32	6	15	Tu, Th 13:00- 14:15
44	Course 33	8	15	W 18:00- 20:30
45	Course 34	8	14	Tu, Th 16:00- 17:15
46	Course 35	8	17	Tu, Th 17:30- 18:45
47	Course 36	3	20	M, W 17:00- 18:15
48	Course 37	3	21	Tu, Th 7:00- 8:15
49	Course 38-1	4	22	M, W, F 10:00- 10:50
50	Course 38-2	4	22	M, W, F 9:00- 9:50
51	Course 39-1	4	23	M, W, F 10:00- 10:50
52	Course 39-2	4	23	M, W, F 11:00- 11:50
53	Course 40	4	23	Tu, Th 10:00- 11:40
54	Course 41-1	4	24	M, W, F 9:00- 9:50
55	Course 41-2	4	15	Tu, Th 11:30- 12:45
56	Course 41-3	4	24	M, W, F 10:00- 10:50
57	Course 42	4	18	Tu, Th 10:00- 11:15
58	Course 43	5	14	Tu, Th 7:00- 8:15
59	Course 44	5	11	M, Tu , W, F 11:00- 11:50
60	Course 45	5	24	M, W, F 16:00- 16:50
61	Course 46-1	6	25	M, W, F 7:00- 7:50
62	Course 46-2	6	25	M, W, F 15:00- 15:50
63	Course 47	5	16	Tu, Th 15:00- 16:40
64	Course 48-1	6	25	Tu, Th 8:30- 9:45
65	Course 48-2	6	25	M, W, F 14:00- 14:50
66	Course 49-1	6	11	M, W 13:00- 14:40
67	Course 49-2	6	11	Tu, Th 13:00- 14:40
68	Course 50-1	6	20	Tu, Th 16:00- 17:15
69	Course 50-2	6	20	Tu, Th 17:30- 18:45
70	Course 50-3	6	26	M , W 17:00- 18:15
71	Course 51-1	6	22	Tu, Th 7:00- 8:15
72	Course 51-2	6	22	Tu, Th 11:30- 12:45
73	Course 51-3	6	22	Tu, Th 10:00- 11:15
74	Course 52	8	18	M, W, F 16:00- 16:50
75	Course 53	8	15	M 18:00- 20:30
76	Course 54-1	8	25	Tu, Th 17:30- 18:45
77	Course 54-2	8	1	Tu, Th 17:30- 18:45
78	Course 55	7	16	Tu, Th 13:00- 14:15
79	Course 56	8	1	Tu, Th 16:00- 17:15
80	Course 57	8	18	Tu, Th 19:00- 20:15

Figure 14: Table with the summary of the published courses offered for Spring 2020.

Accordingly, with the information published at the Catalogue of Courses for Spring 2020 we prepared that information in a table (figure 14) as well as in timetables for easy visualization, the timetables with data from the Catalogue are with darker headline. Then we can analyze the difference between both approaches for the careers and its' semesters:

- **Comparisons for second and third semester for careers 2 and 3:** for second semester of career 3 and career 2 we can see that on the schedules obtained with the model we have all programmed for the morning while at the catalogue we have 2 courses at the evening. As for term 3 of career 3 we can see that we had to split the courses from the catalogue as there is an overlap between course 2 and 5, this can be overlooked as there is different plan, meaning that those are meant for different groups of students, still, we have courses on the afternoon while on the results from the model we have all the courses on the morning.

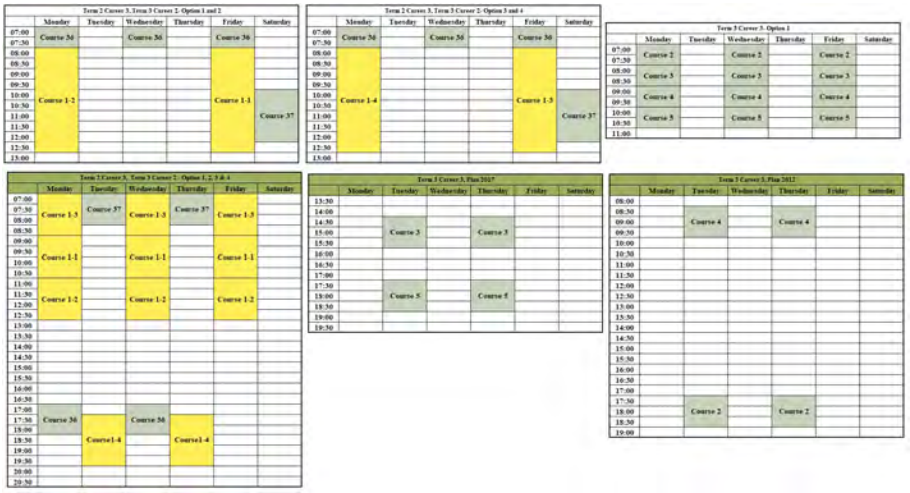


Figure 15: Comparisons for semester 2 and 3 from career 3, and semester 3 of Career 2, with results from integer programming formulation and Catalogue of Courses.

- **Comparisons for semester 4:**
  - *Career 1:* here we don't have problems of overlaps, but we can see that the schedules obtained with integer programming are condensed in 3 days and are similar between options while the courses offered are scheduled in 5 days a week.

Figure 16: Comparisons for semester 4 from career 1 with results from integer programming formulation and Catalogue of Courses.

– *Career 2:* here we do have overlaps, an example can be seen at the schedule offered in the option 2, we are missing course 41 and if we look at the possibilities shown on the other options, the overlap can be seen. Also, the option 3 from the offered courses is the only one with the course 42. If we take a look at the schedules obtained with our model we can see all the options have all the courses needed.

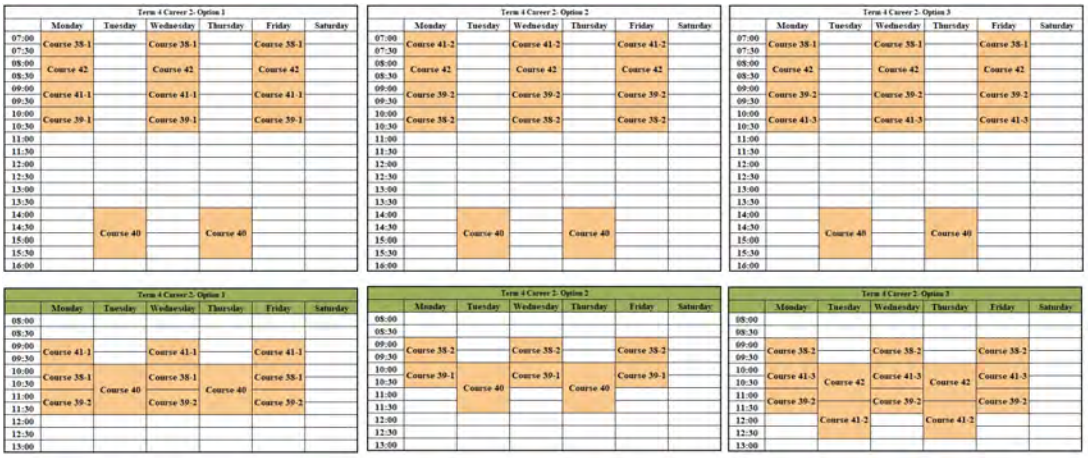


Figure 17: Comparisons for semester 4 from career 2 with results from integer programming formulation and Catalogue of Courses.

– *Career 3:* at this ones, we can see that on the offered courses there was an overlap between courses 10 and 11, with the proposed approach we don't get overlaps.

Figure 18: Comparisons for semester 4 from career 3 with results from integer programming formulation and Catalogue of Courses.

- **Comparisons for semester 5:** we can see for career 1, that there is missing course 44 at the offered courses, also there are some holes in both proposals. For career 2 we can see way more holes between courses at the offered schedule (on Tuesdays and Thursdays), we have 3 classes with lot of free time between them, while at the proposed solution with our model we get one hole on Monday and one on Wednesday which is less than the other approach. Finally, for career 3 the difference can be seen at the number of days we get courses scheduled, at the offered schedule we have 5 days classes while at our approach we have only 3 days with classes.

Figure 19: Comparisons for semester 5 from career 1, 2, and 3 with results from integer programming formulation and Catalogue of Courses.

- **Comparisons for semester 6:**
  - *Career 1:* at the proposed schedules with the integer programming programming approach we can see that option 3 doesn't have Course 47, this is because it was left aside on purpose as we imagined not everyone was meant to take that course, this one is one of the courses shared in career 1 and 2, by having only one section of it, we considered it could be also considered for career one but maybe not all the students of career one would take it. As for the proposals from what was offered we can see the same overlap for option 3. Now, if we look at the holes in between classes we can see that at the timetables obtained with our model, the biggest one is 1 hour while for the offered courses, the biggest hole is of 3 hours. Also we can see that the rule that says that *courses from fifth term and on should be in the afternoon* is not obeyed as we can see courses at 10 hrs., while the earliest start at the timetables obtained with our model is at 13 hrs.

Figure 20: Comparisons for semester 6 from career 1 with results from integer programming formulation and Catalogue of Courses.

  - *Career 2:* here schedules we have something similar, all the proposed timetables have all the necessary courses and as for the holes between courses they are almost even, what we can say is better at the solution with our model is that we are avoiding the 7 am time slot, this means that most of the courses are concentrated at the afternoon.

Figure 21: Comparisons for semester 6 from career 2 with results from integer programming formulation and Catalogue of Courses.

– *Career 3:* what we can first see at this timetables is that for the third option for the offered courses with the catalogue we can't take course 17, as it overlaps with course 14 section 3. This doesn't happen at the timetables obtained with our integer programming model.

Figure 22: Comparisons for semester 6 from career 3 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for terms 7 and 8 from careers 1 and 2:** as we could see at the previous comparison, career 1 and 2 have some similarities in their courses, for that reason we can merge the comparison for these careers for terms 7 and 8.

  For term 7 we can see one timetable that belongs to career 2, this one can be applied to career 1 too as the courses mentioned are also offered to students in career 1. Here we can see that the schedules obtained with the proposed model are concentrated on the evening while the courses offered (signaled with a darker color in the heading of the timetable) are more dispersed. Having a start at 13 hrs we can find it inconvenient if we consider that we are talking of last-year students who may be working at their practices projects.

  For the courses of term 8, we have first the timetables for career 1, by not having the name of the courses it's hard to identify which may be more convenient. As we have previously mentioned, there are some shared courses with career 1, knowing this we can proceed to say that for the proposed timetable with our model, this shared courses can be taken while with the offered courses we have 2 courses that doesn't appear as there might be an overlap.

  Now, for term 8 from career 2, we can see that in both approaches the timetables for option 1 and 2 are similar. We have only one course with 2 sections and these are taught by different teachers making it possible to be scheduled at the same time. Our model makes that assignation as it was done too in the real offered courses, the difference here is that for the solution obtained with our model we get little to none holes between

classes, while at the offered course we have a hole on Wednesdays of 2 hours. Also, the earliest start time at the offered courses is 13 hrs, while at the solution with the model is 16hr. The start at 13hrs might be inconvenient if we recall that we are talking about last-year students.

Figure 23: Comparisons for semesters 7 and 8 from careers 1 and 2 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semesters 7 and 8 from career 3:** first we have the timetables for semester 7 of career 3, we can see it's only one course, this particular course is a control course and we can see here something that wasn't considered as it's part of the confidential information, apparently, for these kind of courses the structure is meant to be scheduled in only one day, aside from it, there is nothing else to discuss with this term. Moving onto the eighth semester for career 3, we can see from start that the proposed timetables with our model, both have all the courses, with its respective variants in sections but complete and are concentrated in the afternoon. When we look at the timetables offered for the term we can see that there are some holes but more importantly, there are overlaps, the evident one is with course 25, it overlaps with course 24 section one but also, there is one overlap that can't be easily seen, we are missing course 29, which was planned to take place on Thursdays from 17:30 to 20:00 hrs. In none of the 2 options it can be placed making it difficult for the students to have this course.

Figure 24: Comparisons for semesters 7 and 8 from career 3 with results from integer programming formulation and Catalogue of Courses.

After making the comparison with the courses for Spring 2020, we also did it with information from Fall 2020. Once again all names of courses and professors are in a generic form. We were going to use same assignation for I (our time slots) but when we were looking at the published schedules for the term we noticed some differences in the way structures were working. We attributed it to the situation with the Covid-19 virus which reached Mexico on March 2020 and made all courses go into online mode. Due to this, we assumed new rules were being implemented. An example was with structure 1, before this situation we could schedule it in 3 sessions per week, but analyzing the data we could see this structure had been replaced with more slots for giving classes only 2 days. We listed some of the rules we could interpret from the catalogue of courses:

- Classes on Saturdays were implemented.
- Now there were slots for two-days-per-week classes for Mondays and Wednesdays, Tuesdays and Thursdays, or Fridays and Saturdays.
- New rules related to terms:
  - Courses from terms 1 to 3 were mostly programmed on Mondays and Wednesdays, or Tuesdays and Thursdays. And most of them on morning slots.
  - Courses from 4 to 6 were programmed mostly on Mondays and Wednesdays, or Tuesdays and Thursdays.
  - Courses of last terms were programmed mostly on Friday and Saturday with occasional courses on Tuesdays and Thursdays.

- We complemented the previous point with the previous rules for assignation (*courses from first four terms on mornings and the rest in the afternoon*). Thinking that for terms from 1 to 3 would be for morning classes, from 4 to 6 in the afternoon and for last-year courses to be in whichever slot on Friday and Saturday.

This is important as those were the guidelines to create our data file, specifically for the IJ parameter, which if we recall relates to the possible time slots to which each course can be assigned, the number of available slots became bigger as there were new possibilities to assign the courses.

With this cleared up, we can proceed to see our results, at figure 25 we can see the summary table with the results from our model, while at figure 26 we have what we could get of information from the Catalogue of courses for Fall 2020.

Course J	Name	Slot I	Term	Professor	Schedule
1	Course 1-1	372	2	1	M, W 10:00- 12:30
2	Course 1-2	375	2	1	M, W 13:00- 15:30
3	Course 1-3	369	2	1	M, W 7:00- 9:30
4	Course 2	14	2	2	Tu, Th 7:00- 8:15
5	Course 3-1	76	3	3	M 10:00- 12:30
6	Course 3-2	77	3	3	M 13:00- 15:30
7	Course 3-3	75	3	3	M 7:00- 9:30
8	Course 4-1	14	3	5	Tu, Th 7:00- 8:15
9	Course 4-2	14	3	4	Tu, Th 7:00- 8:15
10	Course 5-1	15	3	2	Tu, Th 8:30- 9:45
11	Course 5-2	16	3	2	Tu, Th 10:00- 11:15
12	Course 6	19	5	5	Tu, Th 14:30- 15:45
13	Course 7-1	237	5	6	M, W 16:00- 16:50
14	Course 7-2	235	5	6	M, W 14:00- 14:50
15	Course 7-3	236	5	6	M, W 15:00- 15:50
16	Course 8-1	20	5	5	Tu, Th 16:00- 17:15
17	Course 8-2	21	5	5	Tu, Th 17:30- 18:45
18	Course 9	18	4	7	Tu, Th 13:00- 14:15
19	Course 10	114	4	7	Th 16:00- 18:30
20	Course 11	18	5	8	Tu, Th 13:00- 14:15
21	Course 12	236	5	8	M, W 15:00- 15:50
22	Course 13-1	21	5	9	Tu, Th 17:30- 18:45
23	Course 13-2	20	5	9	Tu, Th 16:00- 17:15
24	Course 14	22	5	9	Tu, Th 19:00- 20:15
25	Course 15-1	234	5	10	M, W 13:00- 13:50
26	Course 15-2	237	5	10	M, W 16:00- 16:50
27	Course 16	408	5	9	M, W 17:30- 18:45
28	Course 17-1	235	5	4	M, W 14:00- 14:50
29	Course 17-2	234	5	4	M, W 13:00- 13:50
30	Course 18	406	6	11	M, W 14:30- 15:45
31	Course 19	18	5	12	Tu, Th 13:00- 14:15
32	Course 20	412	7	13	F, Sa 10:00- 11:15
33	Course 21	410	7	5	F, Sa 7:00- 8:15
34	Course 22	413	7	13	F, Sa 11:30- 12:45
35	Course 23-1	244	7	13	Tu, Th 11:00- 11:50
36	Course 23-2	245	7	13	Tu, Th 12:00- 12:50
37	Course 23-3	243	7	13	Tu, Th 10:00- 10:50
38	Course 24	411	7	2	F, Sa 8:30- 9:45
39	Course 25	414	7	3	F, Sa 13:00- 14:15
40	Course 26	22	5	14	Tu, Th 19:00- 20:15
41	Course 27	16	7	15	Tu, Th 10:00- 11:15
42	Course 28-1	410	7	16	F, Sa 7:00- 8:15
43	Course 28-2	411	7	16	F, Sa 8:30- 9:45
44	Course 29	414	7	14	F, Sa 13:00- 14:15
45	Course 30	18	4	17	Tu, Th 13:00- 14:15

Course J	Name	Slot I	Term	Professor	Schedule
46	Course 31-1	19	5	8	Tu, Th 14:30- 15:45
47	Course 31-2	20	5	8	Tu, Th 16:00- 17:15
48	Course 32-1	15	3	18	Tu, Th 8:30- 9:45
49	Course 32-2	14	3	18	Tu, Th 7:00- 8:15
50	Course 32-3	16	3	18	Tu, Th 10:00- 11:15
51	Course 33-1	77	5	6	M 13:00- 15:30
52	Course 33-2	100	5	6	W 10:00- 12:30
53	Course 33-3	76	5	6	M 10:00- 12:30
54	Course 34-1	76	3	19	M 10:00- 12:30
55	Course 34-2	75	3	19	M 7:00- 9:30
56	Course 35-1	21	4	18	Tu, Th 17:30- 18:45
57	Course 35-2	19	4	18	Tu, Th 14:30- 15:45
58	Course 36	20	4	20	Tu, Th 16:00- 17:15
59	Course 37	57	4	15	M, W 15:00- 16:40
60	Course 38-1	19	4	16	Tu, Th 14:30- 15:45
61	Course 38-2	21	4	16	Tu, Th 17:30- 18:45
62	Course 39	55	4	18	M, W 13:00- 14:40
63	Course 40	18	5	15	Tu, Th 13:00- 14:15
64	Course 41	57	5	21	M, W 15:00- 16:40
65	Course 42-1	20	5	22	Tu, Th 16:00- 17:15
66	Course 42-2	19	5	22	Tu, Th 14:30- 15:45
67	Course 43	55	5	23	M, W 13:00- 14:40
68	Course 44-1	21	5	21	Tu, Th 17:30- 18:45
69	Course 44-2	19	5	21	Tu, Th 14:30- 15:45
70	Course 45	21	5	24	Tu, Th 17:30- 18:45
71	Course 46-1	18	5	22	Tu, Th 13:00- 14:15
72	Course 46-2	21	5	22	Tu, Th 17:30- 18:45
73	Course 47-1	20	5	24	Tu, Th 16:00- 17:15
74	Course 47-2	18	5	24	Tu, Th 13:00- 14:15
75	Course 48	410	7	24	F, Sa 7:00- 8:15
76	Course 49	55	6	20	M, W 13:00- 14:40
77	Course 50	19	6	17	Tu, Th 14:30- 15:45
78	Course 51	18	6	25	Tu, Th 13:00- 14:15
79	Course 52	102	6	18	W 16:00- 18:30
80	Course 53	411	7	21	F, Sa 8:30- 9:45
81	Course 54-1	413	7	26	F, Sa 11:30- 12:45
82	Course 54-2	414	7	26	F, Sa 13:00- 14:15
83	Course 55	413	7	16	F, Sa 11:30- 12:45
84	Course 56	16	8	27	Tu, Th 10:00- 11:15
85	Course 57	17	7	23	Tu, Th 11:30- 12:45
86	Course 58	412	7	17	F, Sa 10:00- 11:15
87	Course 59	18	8	23	Tu, Th 13:00- 14:15
88	Course 60	446	7	27	F, Sa 16:00- 17:15
89	Course 61	445	8	27	F, Sa 14:30- 15:45

Figure 25: Summary of the results for the Multiple section instance for Careers 1,2 and 3. Considering all the sections needed at Fall 2020.

Course J	Name	Term	Professor	Schedule
1	Course 1-1	2	1	M, W 8:00- 10:30
2	Course 1-2	2	1	F, Sa 8:00- 10:30
3	Course 1-3	2	1	Tu, Th 8:30- 11:00
4	Course 2	2	2	Tu, Th 10:00- 11:15
5	Course 3-1	3	3	Tu, Th 8:30- 9:45
6	Course 3-2	3	3	Tu, Th 11:30, 12:45
7	Course 3-3	3	3	F 11:00- 13:30
8	Course 4-1	3	4	Tu, Th 13:00- 14:15
9	Course 4-2	3	5	Tu, Th 13:00- 14:15
10	Course 5-1	3	2	Tu, Th 10:00- 11:15
11	Course 5-2	3	2	Tu, Th 17:30- 18:45
12	Course 6	5	5	M, W 17:30- 18:45
13	Course 7-1	5	6	Tu, 10:00- 11:40
14	Course 7-2	5	6	Th 10:00- 11:40
15	Course 7-3	5	6	F 11:00- 12:40
16	Course 8-1	5	5	M, W 14:30- 15:45
17	Course 8-2	5	5	M, W 16:00- 17:15
18	Course 9	4	7	Tu, Th 16:00- 17:15
19	Course 10	4	7	Tu, Th 17:30- 18:45
20	Course 11	5	8	Tu, Th 14:30- 15:45
21	Course 12	5	8	Th 12:00- 13:40
22	Course 13-1	5	9	M, W 11:30- 12:45
23	Course 13-2	5	9	M, W 14:30- 15:45
24	Course 14	5	9	Tu, Th 13:00- 14:15
25	Course 15-1	5	10	Tu 10:00- 11:40
26	Course 15-2	5	10	Th 10:00- 11:40
27	Course 16	5	9	M, W 16:00- 17:15
28	Course 17-1	5	4	Tu, Th 16:00- 17:15
29	Course 17-2	5	4	Tu, Th 18:00- 19:15
30	Course 18	6	11	Tu, Th 13:00- 14:15
31	Course 19	5	12	Tu, Th 14:30- 15:45
32	Course 20	7	13	F, Sa 11:30- 12:45
33	Course 21	7	5	F, Sa 13:00- 14:15
34	Course 22	7	13	Tu, Th 16:00- 17:15
35	Course 23-1	7	13	F 13:00- 14:40
36	Course 23-2	7	13	Tu 17:30- 19:10
37	Course 23-3	7	13	Th 17:30- 19:10
38	Course 24	7	2	Sa 9:00- 11:30
39	Course 25	7	3	F 8:00- 10:30
40	Course 26	5	14	Tu, Th 17:30- 18:45
41	Course 27	7	15	F, Sa 11:30- 12:45
42	Course 28-1	7	16	Tu, Th 16:00- 17:15
43	Course 28-2	7	16	M, W 19:00- 20:15
44	Course 29	7	14	Tu, Th 19:00- 20:15
45	Course 30	4	17	M, W 13:00- 14:45

Course J	Name	Term	Professor	Schedule
46	Course 31-1	5	8	M, W 13:00- 14:15
47	Course 31-2	5	8	Tu, Th 17:30- 18:45
48	Course 32-1	3	18	M, W 13:00- 14:15
49	Course 32-2	3	18	M, W 8:30- 9:45
50	Course 32-3	3	18	M, W 10:00- 11:15
51	Course 33-1	5	6	M 10:00- 12:30
52	Course 33-2	5	6	W 10:00- 12:30
53	Course 33-3	5	6	F 8:00- 10:30
54	Course 34-1	3	19	M, W 8:30- 9:45
55	Course 34-2	3	19	M, W 10:00- 11:15
56	Course 35-1	4	18	Tu, Th 8:30- 9:45
57	Course 35-2	4	18	Tu, Th 10:00- 11:15
58	Course 36	4	20	M, W 11:30- 12:45
59	Course 37	4	15	Tu 19:00- 20:40
60	Course 38-1	4	16	Tu, Th 13:00- 14:15
61	Course 38-2	4	16	M, W 17:30- 18:45
62	Course 39	4	18	M 8:00- 9:40
63	Course 40	5	15	Tu, Th 13:00- 14:15
64	Course 41	5	21	M 8:00- 9:40
65	Course 42-1	5	22	M, W 16:00- 17:15
66	Course 42-2	5	22	Tu, Th 16:00- 17:15
67	Course 43	5	23	Tu 10:00- 11:40
68	Course 44-1	5	21	M, W 14:30- 15:45
69	Course 44-2	5	21	Tu, Th 14:30- 15:45
70	Course 45	5	24	Tu, Th 16:00- 17:15
71	Course 46-1	5	22	M, W 17:30- 18:45
72	Course 46-2	5	22	Tu, Th 16:00- 17:15
73	Course 47-1	5	24	Tu, Th 10:00- 11:15
74	Course 47-2	5	24	Tu, Th 8:30- 9:45
75	Course 48	7	24	Tu, Th 13:00- 14:15
76	Course 49	6	26	M 8:00- 9:40
77	Course 50	6	17	Tu, Th 7:00- 8:15
78	Course 51	6	25	Tu, Th 17:30- 18:45
79	Course 52	6	18	Tu, Th 13:00- 14:15
80	Course 53	7	21	F, Sa 11:30- 12:45
81	Course 54-1	7	26	Tu 17:30- 20:00
82	Course 54-2	7	26	Th 17:30- 20:00
83	Course 55	7	16	Tu, Th 17:30- 18:45
84	Course 56	8	27	M, W 17:30- 18:45
85	Course 57	7	23	F, Sa 8:30- 9:45
86	Course 58	7	17	F, Sa 10:00- 11:15
87	Course 59	8	23	F, Sa 13:00- 14:15
88	Course 60	7	27	Sa 14:00- 16:30
89	Course 61	8	27	Sa 17:00- 19:30

Figure 26: Table with the resume of the offered courses for Fall 2020.

We then proceeded to compare each term as it was done for Spring:

- **Comparison for semesters 2 and 3 for Career 3:** for term 2 we can see that on the offered courses if we choose to take course 1 at section 3 we won't be able to take course 2, this doesn't happen at the schedule proposed with the integer programming model. As for term 3 the main difference can be the hole between course 4 (in either section 1 or 2) and course 5 section 2, it's 3 hours between those in the offered courses while the biggest hole at the other proposal is at most 1 hour if we choose the named courses.

Figure 27: Comparisons for semesters 2 and 3 from career 3 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semester 3 for Career 2:** at this comparison we can see that for the proposed schedule obtained with our model, we have all the sections free, without overlaps, giving flexibility to choose, meanwhile, at the offered courses we can see some overlaps, but the main problem is that there is no way to put course 32 section 3 as it overlaps.

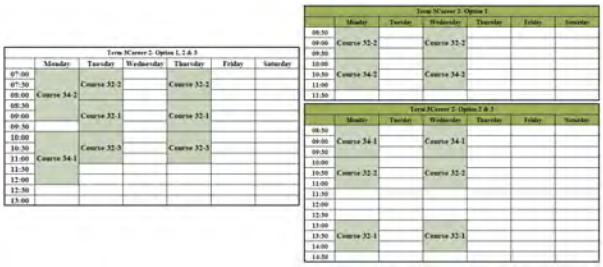


Figure 28: Comparisons for semester 3 from career 2 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semester 4 for Career 2:** we have this divided into plan 2012 and plan 2017 as there were some differences into the sections and courses.
  - *Plan 2012:* here we can see a more compressed schedule at the solution given by our model. Both proposals have all the courses needed but the offered courses at the catalogue have big holes in between.

Figure 29: Comparisons for semester 4 from career 2 for plan 2012 with results from integer programming formulation and Catalogue of Courses.

  - *Plan 2017:* here we can see that semester 4 from plan 2017 of career 2 shares course 1 of term 2 of career 3. While with the proposals obtained with the model, the students can choose either of the 3 sections for course 1, at the offered courses they can only choose 1 or 2. As for the other courses, we can see that with the integer programming program, both options of courses are visually the same, each with different combinations for the sections while at the offered schedules have big differences as can be seen with course 38, if by any chance a student

enrolled in section 1 of course 35 and in section 2 of course 38, that would mean he or she would have a hole of 7 hours and a half.

Figure 30: Comparisons for semester 4 from career 2 for plan 2017 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semesters 4 and 6 for career 3:** here we can see the possible case in which the offered courses found a better arrangement, we have course 9 and 10 close while at the proposal obtained with the model we have a hole of an hour and a half, as for the course 18 in the proposal with our model we have it in different days making the students to go to university 4 days a week and with the offered courses they would go 2 days a week.

Figure 31: Comparisons for semesters 4 and 6 from career 3 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semester 5 for career 1:** at the options obtained with our model, we have that students have classes 2 days a week while with the offered courses they have 4 days a week classes. Also the biggest hole at both options obtained with integer programming programming is an hour and a half as for the offered courses we can even have a hole of 6 hours.

Figure 32: Comparisons for semester 5 from career 1 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semester 5 for career 2:** we have this divided into plan 2012 and plan 2017 as there were some differences into the sections and courses, some courses are taken by both plans as there are courses with same schedule and content but with different name (which is disclosed due confidentiality)
  - *Plan 2012:* both options obtained with integer programming are complete with the courses needed and no big holes in between, at most half an hour. As for the offered courses, we have at option 5 an overlap between courses 42 section 2 and course 45.

Figure 33: Comparisons for semester 5 from career 2 for plan 2012 with results from integer programming formulation and Catalogue of Courses.

  - *Plan 2017:* with the options obtained with integer programming we have students that need to go 3 days a week to the university in any option they choose, in the other hand, with the set of offered courses we can see that students can end up having classes 5 days a week. Also, the main variation between the schedules obtained with the model is

the day they have course 33, as for the schedules offered we have big differences between the options.

Figure 34: Comparisons for semester 5 from career 2 for plan 2017 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semester 5 for career 3:** we have this divided into plan 2012 and plan 2017 as there were some differences into the sections and courses.
  - *Plan 2012:* we can see that for the offered courses there are some missing courses as there are overlaps, at option 1 we are missing course 14, and at option 2 we are missing courses 12 and 16.

Figure 35: Comparisons for semester 5 from career 3 for plan 2012 with results from integer programming formulation and Catalogue of Courses.

  - *Plan 2017:* we have also some overlaps for the offered courses like course 16 that doesn't appear at option 2 as it overlaps with course 8 section 2. In terms of spaces between courses, it can be seen that is similar for the offered courses and the obtained with integer programming.

Figure 36: Comparisons for semester 5 from career 3 for plan 2017 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semester 6 for career 2:** the first advantage we can see with the proposed schedule obtained with IP is that it is centered in the afternoon, not as dispersed like the courses offered for the term.

Figure 37: Comparisons for semester 6 from career 2 with results from integer programming formulation and Catalogue of Courses.

- **Comparison for semesters 7 and 8:** we merged terms 7 and 8 because we have few courses of term 8 and those can be mixed with term 7 courses. Also, as career 1 and 2 are similar in some courses for this terms they are together and after them is the comparison for career 3.
  - *Career 1 and 2:* for both careers we have option 1 and 2 in the same timetable as the courses that have different sections happen to don't overlap due to they being taught by the same professor and making it not feasible to be overlapped. For both careers, the best option in schedules can be the ones proposed by the IP model, they lower the days of classes to 4, in the other hand, with the offered assignation of classes, students have classes 6 days a week.

Figure 38: Comparisons for semesters 7 and 8 from careers 1 and 2 with results from integer programming formulation and Catalogue of Courses.

- *Career 3:* if we take a look at the courses offered, we can see there is an overlap between courses 21 and 60, aside from that we face that they have more hole between courses.

Figure 39: Comparisons for semesters 7 and 8 from career 3 with results from integer programming formulation and Catalogue of Courses.

These are the results for the comparisons between what was the output for the programming of courses on a real instance and actual method and what could be obtained with the proposed IP model.

5.5 Instances solutions with Constraint programming.

For this approach's model, the code used can be seen at figure 40 changing the name of the data file that will be used for each instance that wants to be solved. Using this, we ran 2 instances:

- With data of the 3 careers, using all the needed groups for the term of Spring 2020.
- With data of the 3 careers, using all the needed groups for the term of Fall 2020.

Figure 40: Code used for the Constraint Programming approach.

First, for the data of Spring, is the same instance we used for the Integer programming approach. For this model we needed to build the data file with our two dynamic arrays (INCOMP and STRUCT), we needed to list up all the courses to know which teacher and semester each belonged to (for de INCOMP array) and also the structure of the course to build the STRUCT array.

From the model we obtained the listing of all the courses scheduled as it can be seen at figure 41. There we can also view that first the model prints the first domains for all the variables and right after applying the constraints of ”Structure” it prints the new domain for each variable.

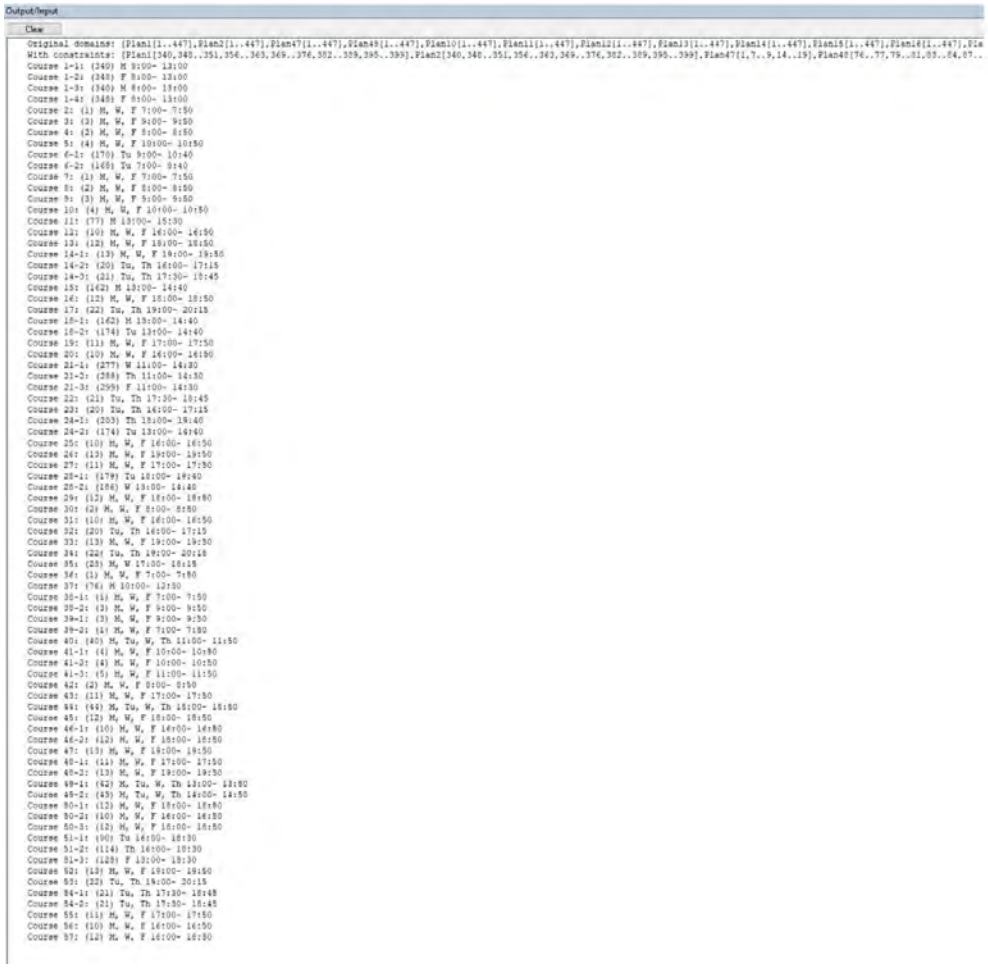


Figure 41: Results obtained with the constraint programming approach for the instance of Spring 2020.

Translating this into timetables, we could easily see there were no overlaps between the courses. We can see the summary of timetables at figure 42.

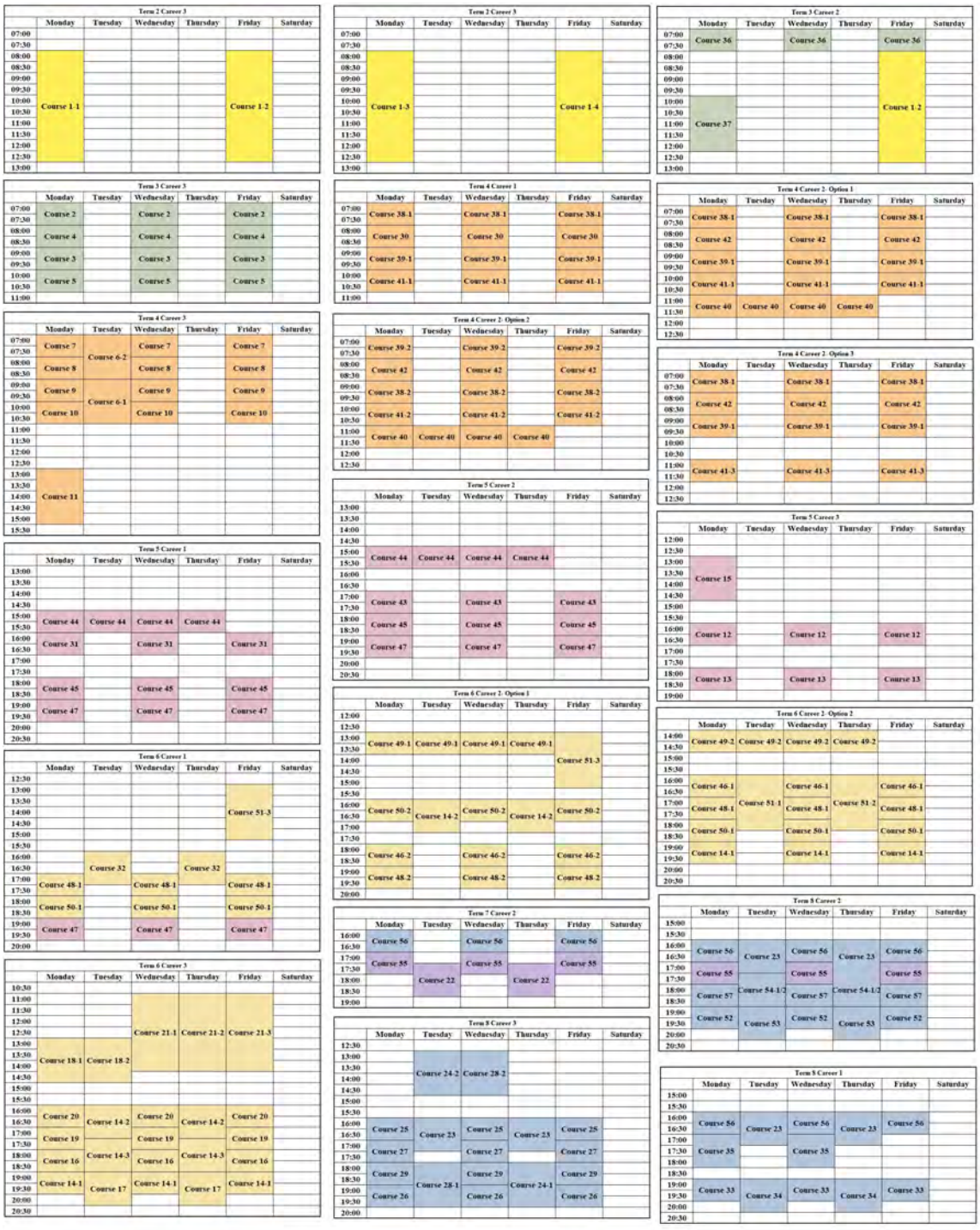


Figure 42: Summary of timetables obtained with the constraint programming approach for the instance of Spring 2020.

As done with the integer programming approach, we divided the timetables to compare and view some of the differences between what was obtained from the

constraint programming model and the published information in the Catalogue of Courses, this last is presented in timetables with darker headlines.

- **Comparisons for second and third semester for careers 2 and 3:** we can see that the courses of second semester have different assignations, having in the constraint programming approach what seems to be overlaps but this courses are taught by different teachers which means that instead of using 4 slots for 4 courses we have 2 slots used for the same 4 courses. For the courses of career 2 for semester 3 we have no big overlaps but certainly not a huge gap in between as what was offered. As for term 3 for career 3 we can see that for the offered courses there is an overlap between courses 2 and 5, this could be ignored as those may be from different plans meaning there is no big problem but with the solution with CP we have no overlaps and the difference is in terms of having to start earlier or finish later.

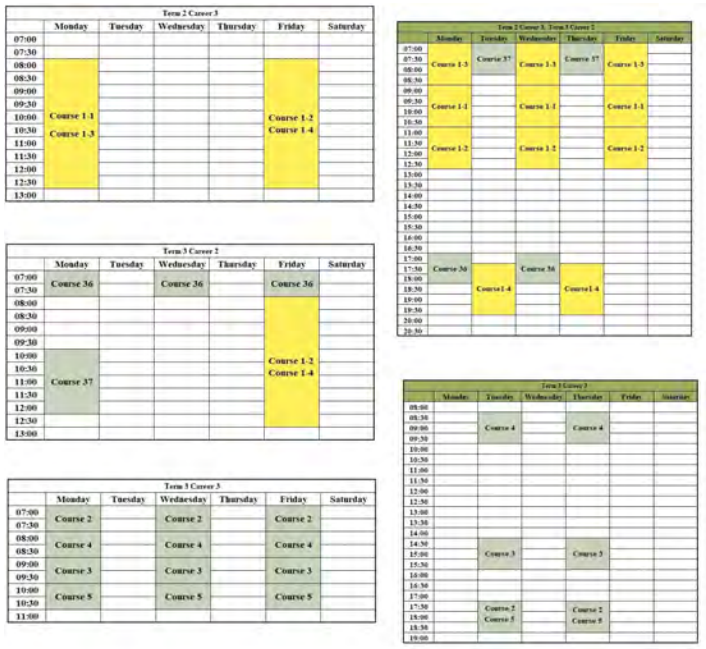


Figure 43: Comparisons for semester 2 and 3 from career 3, and semester 3 of Career 2, with results from constraint programming formulation and Catalogue of Courses.

- **Comparisons for semester 4:**
  - *Career 1:* the main difference is the number of days with courses scheduled, with the constraint programming solution we have classes only 3-days a week. This can vary for the different groups for the classes but a suggestion like this is viable.

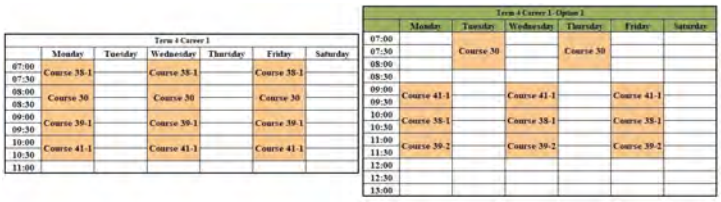


Figure 44: Comparisons for semester 4 from career 1 with results from constraint programming formulation and Catalogue of Courses.

– *Career 2:* here we have displayed all the different chances of schedules. We can see no big problems for the CP approach and it cannot be said same from the offered schedules as we have from the 3 options 2 with incomplete set of courses to take.

Figure 45: Comparisons for semester 4 from career 2 with results from constraint programming formulation and Catalogue of Courses.

– *Career 3:* at this schedules we can view that for the offered courses we have an overlap for courses 10 and 11, this overlap doesn't happen at the Constraint programming schedule.

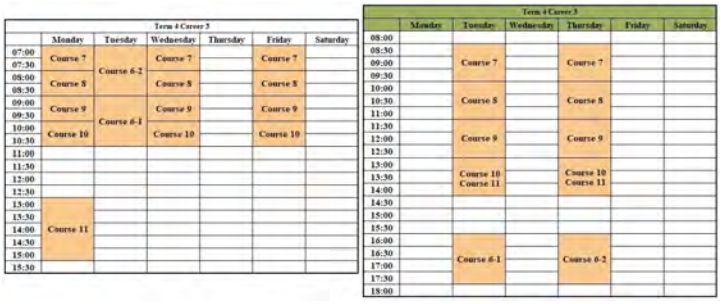


Figure 46: Comparisons for semester 4 from career 3 with results from constraint programming formulation and Catalogue of Courses.

- **Comparisons for semester 5:** at career 1, that there is missing course 44 at the offered courses. For career 2 we can see big spaces between courses at the offered schedule, while at the proposed solution we get the classes carried one after the other with a space for only two days. Finally, we can see that both schedules are similar but at the proposal with CP we get a schedule which uses only 3 days.

Figure 47: Comparisons for semester 5 from career 1, 2, and 3 with results from constraint programming formulation and Catalogue of Courses.

- **Comparisons for semester 6:**
  - *Career 1:* we can see a more compressed schedule with the suggested by Constraint Programming. There are classes scheduled on 5 days but we have the classes set on the afternoons which gives time on the morning for other activities.

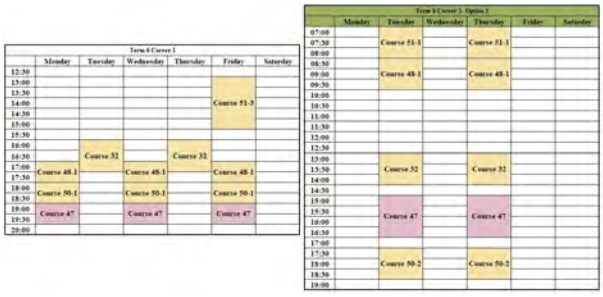


Figure 48: Comparisons for semester 6 from career 1 with results from constraint programming formulation and Catalogue of Courses.

  - *Career 2:* for the proposed schedules we have that the sections can be fit in two timetables but doesn't imply overlaps not allowed. All the classes are focused on the afternoon and not with big spaces between them as what was offered.

Figure 49: Comparisons for semester 6 from career 2 with results from constraint programming formulation and Catalogue of Courses.

  - *Career 3:* while at the offered courses in the catalogue have some overlaps that makes students choose over two courses to take, with the solution of CP we have no overlaps of this type making possible for students to take any schedule they need.

Figure 50: Comparisons for semester 6 from career 3 with results from constraint programming formulation and Catalogue of Courses.

- **Comparison for terms 7 and 8:** as we could see at the previous comparison, career 1 and 2 have some similarities in their courses, for that reason we can merge the comparison for these careers for terms 7 and 8.
  - *Careers 1 and 2:* for semester 7 we have a schedule concentrated in the evening from 18 to 19 hours while the offered schedule on the catalogue has it more expanded. Same can be seen for the schedules for semester 8 of careers 1 and 2, having few spaces between courses and complying with no overlaps of the courses.

Figure 51: Comparisons for semesters 7 and 8 from careers 1 and 2 with results from constraint programming formulation and Catalogue of Courses.

– *Career 3:* for the course of semester 7 we can view that the course has a different structure due to maybe it being a control course and a specification not known, as for the semester 8 schedule we can see a nicer schedule proposed by the constraint programming as it is focused at the afternoon and has few holes between classes.

Figure 52: Comparisons for semesters 7 and 8 from career 3 with results from constraint programming formulation and Catalogue of Courses.

We repeated the process with the data for Fall as we did for the integer programming approach. The code, as mentioned before, is the same, only modifying the name of data file to be used. For the results, we can take a look at the listed scheduled courses in figure 53 and the summary of timetables at figure 54.

Figure 53: Results obtained with the constraint programming approach for the instance of Fall 2020.

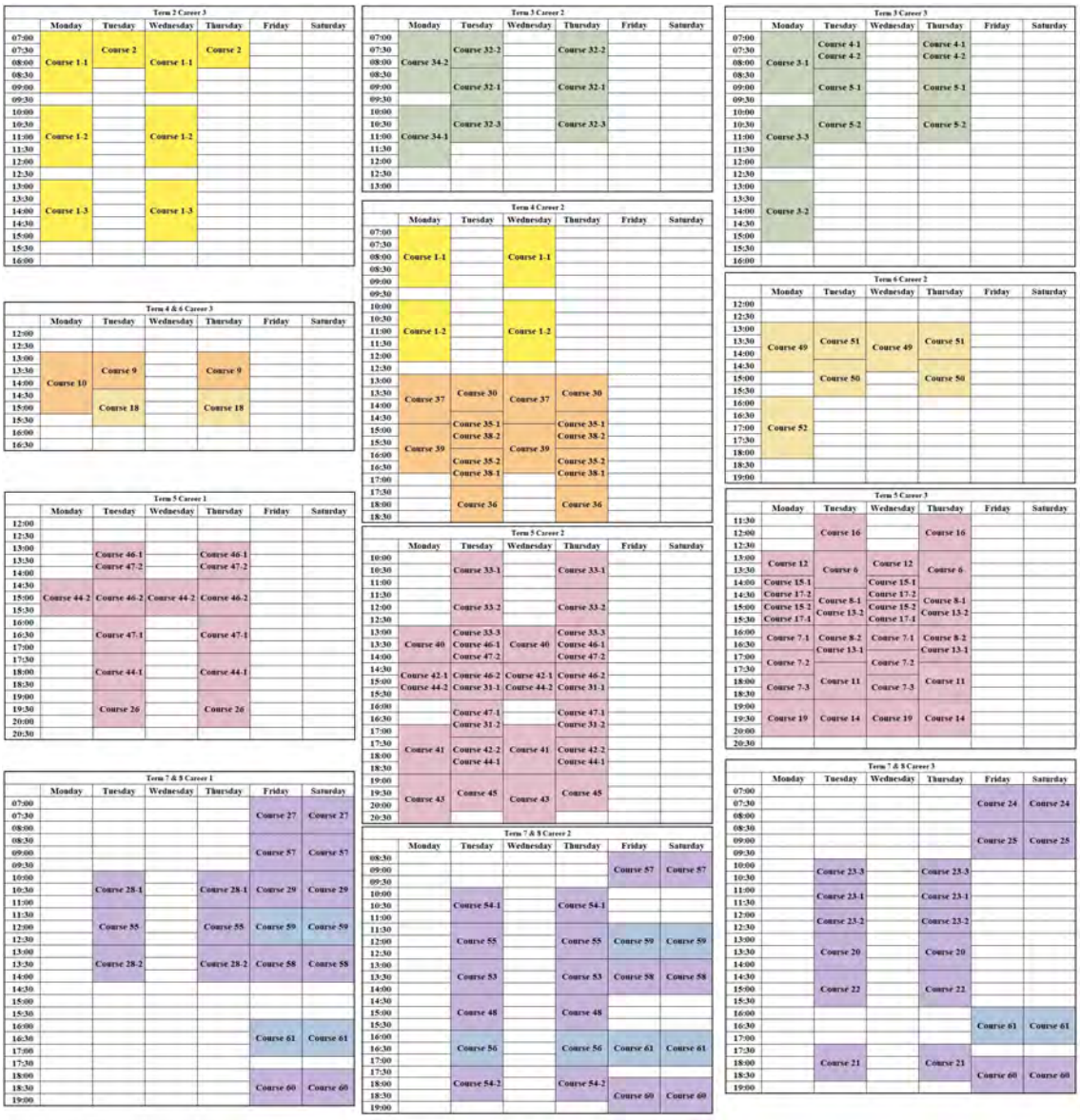


Figure 54: Summary of timetables obtained with the constraint programming approach for the instance of Fall 2020.

This information was compared too to what was offered in named semester. The table with the offered courses in the catalogue can be seen at the previous section and the comparison by semesters is presented below, remember that offered courses from the catalogue are presented in the timetables with darker colour in the headlines:

- **Comparisons for second and third semester for careers 2 and 3:** at term 2 we can see no overlaps on the courses like it can be seen at the offered courses on the catalogue. As for the courses for third semester of career 3,

we can see students have to go 3 days a week with the courses focused on the morning while at the offered ones we have one course in the evening. At the courses of career 2 for semester 3 we have something similar.

Figure 55: Comparisons for semester 2 and 3 from career 3, and semester 3 of Career 2, with results from constraint programming formulation and Catalogue of Courses.

- **Comparisons for semesters 4 and 6:**
  - *Career 2:* at career 4 we can see at the offered courses an overlap for course 1 section 1 and course 39 leaving students with the only choice of group 2. Also, all the courses are extended in all day while the courses offered with Constraint programming are focused at the afternoon. For semester 6 is the same, while what was offered has classes at 7 hours and last one is at 17:30 hours, the suggested schedules with the constraint programming approach are centered in the evening and with no big wholes between them.

Figure 56: Comparisons for semesters 4 and 6 from career 2 with results from constraint programming formulation and Catalogue of Courses.

– *Career 3:* for this set of courses we can see an advantage at the offered courses that students have to go 2 days a week while with the solution proposed with CP they have to go 3 days a week. Nonetheless, an advantage from the CP proposal is that there are no free-hours between classes making the schedule to be compressed. Also, let's note that here semesters 4 and 6 are mixed together due the small number of courses that were needed.

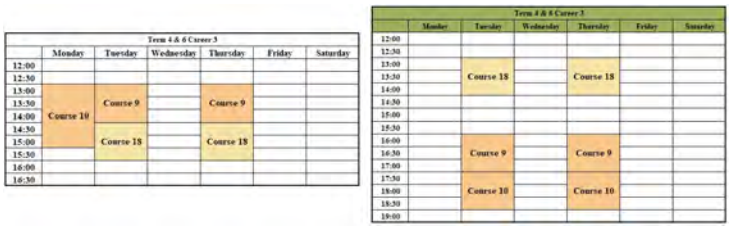


Figure 57: Comparisons for semesters 4 and 6 from career 3 with results from constraint programming formulation and Catalogue of Courses.

- **Comparisons for semester 5:** for career 1 we can see a more compressed schedule, most of the courses in 2 days at the proposal of constraint programming. At career 2 is viewed that courses are scheduled in 4 days while at the offered by the catalogue are in 5 days and all what seems like overlap are courses with multiple groups which allows to take all what is needed. Finally, at career 3 we can see something similar to what happened at career 2.

Figure 58: Comparisons for semester 5 from career 1, 2, and 3 with results from constraint programming formulation and Catalogue of Courses.

- **Comparison for terms 7 and 8:**
  - *Career 1:* what is more notorious is the number of days with courses, the constraint programming approach has only 4 days with courses. None has overlaps and most classes are in the morning.

Figure 59: Comparisons for semesters 7 and 8 from career 1 with results from constraint programming formulation and Catalogue of Courses.

  - *Career 2:* This solution with Constraint programming is similar to the

one of career 1, but we can see that most of the courses are scheduled in Tuesday and Thursday but with few spaces between classes.

Figure 60: Comparisons for semesters 7 and 8 from career 2 with results from constraint programming formulation and Catalogue of Courses.

– *Career 3:* at the proposed schedules we have an overlap within course 21 and course 23 in group 1, on the other hand at the proposed schedule with constraint programming there are no overlaps.

Figure 61: Comparisons for semesters 7 and 8 from career 3 with results from constraint programming formulation and Catalogue of Courses.

The times used for obtaining the answer are small, in average less than 1 second. The results obtained can be perceived as satisfactory as they fulfill the needed constraints.

6 Conclusions

In review, we have used two approaches to find a solution to the every semester problem of assigning courses to a schedule for three university undergraduate programs. The approaches used were an integer programming and a constraint programming model. This approaches were chosen as we found them more suitable given the specific problem we faced. Each of them had its' model created and coded for them to be tested for each of our interest instances.

For the integer programming, we had no objective value but used binary variables to assign for each course a schedule avoiding to have overlaps over courses belonging to the same term and taught by the same teachers, as well as ensuring each of the courses had assigned a schedule. On the other hand, we had the constraint programming approach in which we used a set of constraints to delimit the domain of the variables, another set of constraints was used to state if two courses couldn't be scheduled at the same time. For each of the approaches we created the data files with information of each instance to test or study.

After working on each proposal and having obtained good results for each of them, we can say both are good approaches to help lighten up the work that has to be done for each term in the university. With both approaches, we can have less mistakes by overlapping classes and can get more attractive schedules for the students.

Nonetheless, while we are easing the work of assigning for each course a schedule, we need to consider that the time consuming part for using one of the approaches would be the generation of the data. For sure, we have all the facts for each of the course but still we need to create the different arrays as we may have variations for each term in the courses to be offered, the number of groups for each course and even the new university policies, if something happens to change (as happened this year with the worldwide contingency). Taking this into account we can say that the data collection for the constraint programming approach is simpler than for the integer programming approach.

For the integer programming approach, the hardest part to be created for the data file was the IJ parameter. This parameter was an array of sets of integers. The number of rows corresponds to the number of courses to be assigned and for each row we had to list all the possible slots where the course could be scheduled. Unless we have a change into the policies of the university, we might be able to have an easier creation of the data file after a whole year. In a year, all courses from the career are supposed to have been offered, this means that for the second year using the IP approach we will have a list of all the courses offered with the slots that each can use and for creating the new data file we would only need to add each of the sets for the courses to be assigned. If there are changes this might not work leaving us to work the data file as if it was the first time we are creating it.

As for the constraint programming approach, we had two dynamic arrays to

be created, these dynamic arrays were used to speed up the creation and reading of the data file. Still the creation of both were the most time consuming part of the approach. If we are to write each of the pairs for each dynamic array we would need a lot of time and might lead to some confusions, duplication or missing pairs. To avoid that, an excel document was created in which we had an input sheet to fill either if there existed an incompatibility or if a particular course could be scheduled at a certain slot (for the Structure array). The information of that sheet was later transformed in another sheet into the pairs ready to be translated into a .txt file to be read on the software. This made easier to create the data file and to edit it if needed.

As for the results obtained with each of the approaches, we see no big difference as from a qualitative point of view and considering the constraints we are working with both come up with good results. In summary, both approaches seem to have good solutions and be at the same level of difficulty when building the data file needed. One advantage of constraint programming is that the solver easily can provide several feasible solutions for the problem.

We have proved that for our case, an integer programming and a constraint programming approach can be useful. We are using it for assigning only to each course a schedule as we don't have to worry about nor the assignation of classrooms, the teachers or the groups as this is information previously given. Both approaches work with different sizes of instances and the computational time always is small.

It can be tried to make bigger instances, as seen, some of the courses were shared between careers from the department, this may happen too between departments, meaning there might be more restrictions for some of the courses. By expanding the study to more than one department, we can test if the approaches sustain its' good performance. Also, for getting more insight into which of the approaches gives a better assignation of courses, we would need to find a way to gather more information to create a value which would help us ponder which of the approaches gives a better assignation.

References

Abdullah, S., Burke, E. K., & McCollum, B. (2007). Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In *Metaheuristics* (pp. 153–169). Springer.

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

Badoni, R. P., Gupta, D., & Mishra, P. (2014). A new hybrid algorithm for university course timetabling problem using events based on groupings of students. *Computers & industrial engineering*, *78*, 12–25.

Bolaji, A. L., Khader, A. T., Al-Betar, M. A., & Awadallah, M. A. (2014). University course timetabling using hybridized artificial bee colony with hill climbing optimizer. *Journal of Computational Science*, *5*(5), 809–818.

Cardoen, B., Demeulemeester, E., & Beliën, J. (2010). Operating room planning and scheduling: A literature review. *European journal of operational research*, *201*(3), 921–932.

Daskalaki, S., & Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, *160*(1), 106–120.

Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, *153*(1), 117–135.

Gunawan, A., Ng, K. M., & Poh, K. L. (2007). Solving the teacher assignment-course scheduling problem by a hybrid algorithm. *International Journal of Computer, Information, and Systems Science, and Engineering*, *1*(2), 136.

Gunawan, A., Ng, K. M., & Poh, K. L. (2012). A hybridized lagrangian relaxation and simulated annealing method for the course timetabling problem. *Computers & Operations Research*, *39*(12), 3074–3088.

Hernández Campos, R. A. (2008). Programación de horarios de clases y asignación de salas en la facultad de ingeniería de la universidad diego portales.

Lach, G., & Lübbecke, M. E. (2008). Optimal university course timetables and the partial transversal polytope. In *International workshop on experimental and efficient algorithms* (pp. 235–248).

MirHassani, S. (2006). A computational approach to enhancing course timetabling with integer programming. *Applied Mathematics and Computation*, *175*(1), 814–822.

Pongcharoen, P., Promtet, W., Yenradee, P., & Hicks, C. (2008). Stochastic optimisation timetabling tool for university course scheduling. *International Journal of Production Economics*, *112*(2), 903–918.

Rossi, F., Van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Saldaña Crovo, A., Oliva San Martín, C., & Pradenas Rojas, L. (2007). Modelos de programación entera para un problema de programación de horarios para universidades. *Ingeniare. Revista chilena de ingeniería*, *15*(3), 245–259.

Schimmelpfeng, K., & Helber, S. (2007). Application of a real-world university-course timetabling model solved by integer programming. *Or Spectrum*, *29*(4), 783–803.

Teoh, C. K., Wibowo, A., & Ngadiman, M. S. (2015). Review of state of the art for metaheuristic techniques in academic scheduling problems. *Artificial Intelligence Review*, *44*(1), 1–21.

Xpress, F. (2014). Xpress-kalis user guide. In *User guides* (pp. 1–17).