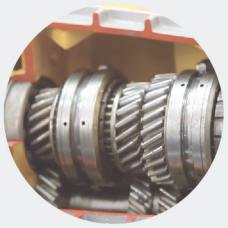
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL






DEM TIMETABLING PROJECT ? DEVELOPMENT/ IMPLEMENTATION OF AN ALGORITHM TO SUPPORT THE CREATION OF TIMETABLES

INÊS MANUELA AFONSO MARRÃO
outubro de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

AN ALGORITHM TO SUPPORT THE CREATION OF TIMETABLES IN ISEP-DEM

Inês Manuela Afonso Marrão

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

Instituto Superior de Engenharia do Porto

AN ALGORITHM TO SUPPORT THE CREATION OF TIMETABLES IN ISEP-DEM

Inês Manuela Afonso Marrão

Student no. 1170486

Dissertation presented to Instituto Superior de Engenharia do Porto to fulfill the necessary requirements to obtain a master's degree in industrial and management Engineering, carried out under the guidance of Professor Manuel Pereira Lopes and co-supervision of Professor João Augusto Bastos.

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

ACKNOWLEDGEMENTS

A work of this nature could not have been carried out without the contribution of some entities and people, who motivate my sincere thanks.

I would like to start by thanking Professor Manuel Pereira Lopes, who agreed to be my supervisor, a deep thank you for all the scientific knowledge he gave me and also for the rigor in guiding this project, which allowed me to grow not only as a professional, but also as a person, as well as Professor João Bastos, my co-advisor. It was in their advice that I found peace of mind and motivation to face this challenge. To Instituto Superior de Engenharia do Porto for being my second home in the past years.

To my family, who walked with me through this process, I thank all the affection and understanding, which would allow me to find balance. To my parents, for the right words and constant encouragement. To my sisters for their complicity and for being there when I needed them most. To my grandmother for always motivating me to do more and better.

To my best friends, I thank them for their patience, the friendly and safe word that comforted and encouraged me to complete this work.

intentionally blank page

ABSTRACT

This work presents the development of an algorithm to support the process of creating academic timetables, specifically aimed at solving the University Course Timetabling Problem. To date, this problem is solved manually in *Instituto Superior de Engenharia do Porto*, where professors and engineers face the complex task of creating timetables based on schedules from previous years.

The proposed solution aimed to support the process of creating timetables at ISEP, reducing the time and human resources required for this task. The developed algorithm uses an integer programming approach and can consider a variety of constraints and preferences of both faculty and students. It was designed to adapt and optimize the timetable creation process as needs evolve, ensuring future demands can be easily accommodated.

The algorithm implementation was based on the Python programming language and the Pyomo library, offering a flexible and efficient approach to optimizing resource allocation. Additionally, the system is designed to import data from real-world sources, simplifying the integration of crucial information.

The result assigned all the 128 one-hour classes among the week, presenting the faculty member, the classroom assigned and the type of class according to each course. This research presents feasible solutions that need improvement on the demanding conditions and restrictions imposed by ISEP. The computational results obtained offered a significantly decrease in the time resource used, compared to the manual work previously done.

KEYWORDS

University Course Timetabling, Integer Programming, Python, Pyomo.

intentionally blank page

INDEX

INDEX OF FIGURES

intentionally blank page

INDEX OF TABLES

intentionally blank page

LIST OF SYMBOLS AND ACRONYMS

List of Acronyms

CBR	Case-Based Reasoning
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
DEM	Mechanical Engineering Department
DF	Distance to Feasibility
GA	Genetic Algorithm
IP	Integer Programming
IPGALS	Improved Parallel Genetic Algorithm and Local Search
ISEP	*Instituto Superior de Engenharia do Porto*
ITC	International Timetabling Competition
LS	Local Search
MIP	Mixed-integer Programming
OR	Operational Research
OT	Tutorial guidelines classes
P.Porto	*Instituto Politécnico do Porto*
PL	Practical-Laboratory classes
S	Seminars
T	Theoretical classes
TP	Theoretical-practical classes
UCTP	University Course Timetabling Problem

intentionally blank page

1. INTRODUCTION

This chapter aims to briefly describe the context and the purpose that backs the development of this project. After that, follows the research question and its objectives, referring concisely to the methods and approaches used.

1.1. Research problem, context, and relevance

This project was developed at *Instituto Superior de Engenharia do Porto* (ISEP), which is one of the schools of the Polytechnic of Porto (P. Porto).

ISEP is an engineering school with at least 13 bachelor's and 15 master's degrees. It has 9 departments, and this project was developed to be implemented firstly at the Mechanical Engineering Department (DEM).

The project consists of developing an algorithm to support the timetabling construction process, which means that the University Course Timetabling Problem (UCTP) must be solved. The UCTP until the present day is solved manually at ISEP – professors and engineers are responsible for this task, which is considered quite complex and most of the time based on schedules provided in the previous years.

Briefly, the current university course timetable of the Mechanical Engineering Department has 5 different kinds of classes: theoretical (T), theoretical-practical (TP), practical-laboratory (PL), tutorial guidelines (OT), and seminars (S). Regarding theoretical classes (T), the faculty is responsible for choosing the weekly distribution of the teaching load. TP and OT classes can be taught for several professors/members of the faculty. PL classes might require the use of specific material, which means the classroom must be prepared to receive the class. There are some classrooms of the department that are used by other departments, considering their availability of them.

To solve this problem, according to Aziz & Aizam (2018), it is essential to consider hard and soft constraints, while allocating the events and resources (human resources, and physical resources, among others). The notorious difference between hard and soft constraints is that the hard ones "should not be violated under any conditions" and the soft ones must be "satisfied as much as possible". The biggest challenge of this problem is solving it by minimizing the conflicts regarding rooms, classes, faculty, and students. Its complexity is also increased due to considerable finite sets, such as the days of the lectures and the period of the day, the class and its characteristics, and the resources needed for each course, among others. The UCTP is considered solved when "a timetable can fulfil the user preferences while considering all the availability and conflict of the resources".

Based on the work developed by Gülcü & Akkan (2020), timetabling problems in universities can be difficult to approach due to the diversity of criteria of each institution, so the authors inscribed a specific problem – curriculum-based university course timetabling problem. Usually, what is observed in most universities is that timetabling is a task performed based on the timetables prevenient from previous years, considering in their majority the preferences of the faculty. The main problem is related to possible requirements for changes, which some algorithms may simplify by optimizing the model again, considering the new constraints added, ensuring that "the changes to the initial timetable are kept to a minimum".

The advocated solution aims to fully support the creation of the design of timetables at ISEP, decreasing the time dispended on this assignment and reducing the human resources used to solve this problem. The algorithm developed is expected to be able to consider potential constraints and restrictions that might be required or wanted in the future, either from the faculty or from the students, adapting and optimizing the process itself.

1.2. Research question and its objectives

Considering the problem mentioned above, the developed work aims to answer the following research question: how can the development of an algorithm be effective in supporting the construction of timetables, in a university context, reducing resources and time used?

In this way, the major goal is to develop and implement a new algorithm that satisfies the demanding conditions and restrictions, accomplishing the following objectives:

- Conduct a comprehensive analysis of current restrictions and obligations.
- Characterize the existing decision variables and constraints in the environment where the algorithm is being applied.
- Analyse previous models implemented and identify similarities with ISEP.
- Choose the best model or method and adapt it to the reality of the project.
- Implement the algorithm in DEM (Mechanical Engineering Department).
- Evaluate the results and compare them to the existing timetables.
- Test the algorithm for further changes/requirements.

1.3. Methodological options

The procedures utilized for the analysis of the various realities encountered in the course timetabling problem had to apprehend the multiple interactions that happen in the given context.

The research structure was established by gathering information about the proposed problem and its objectives, which resulted in the necessary steps for its development. The first step is related to the project's overall plan: the main definitions and goals. Steps 2 and 3 mention the analysis conducted through the literature review, which includes the comparison between the articles studied and the case study reality. The last step includes the adaptation of the model according to the model that was chosen previously. The correspondence between the research and its objectives is shown in the next figure.

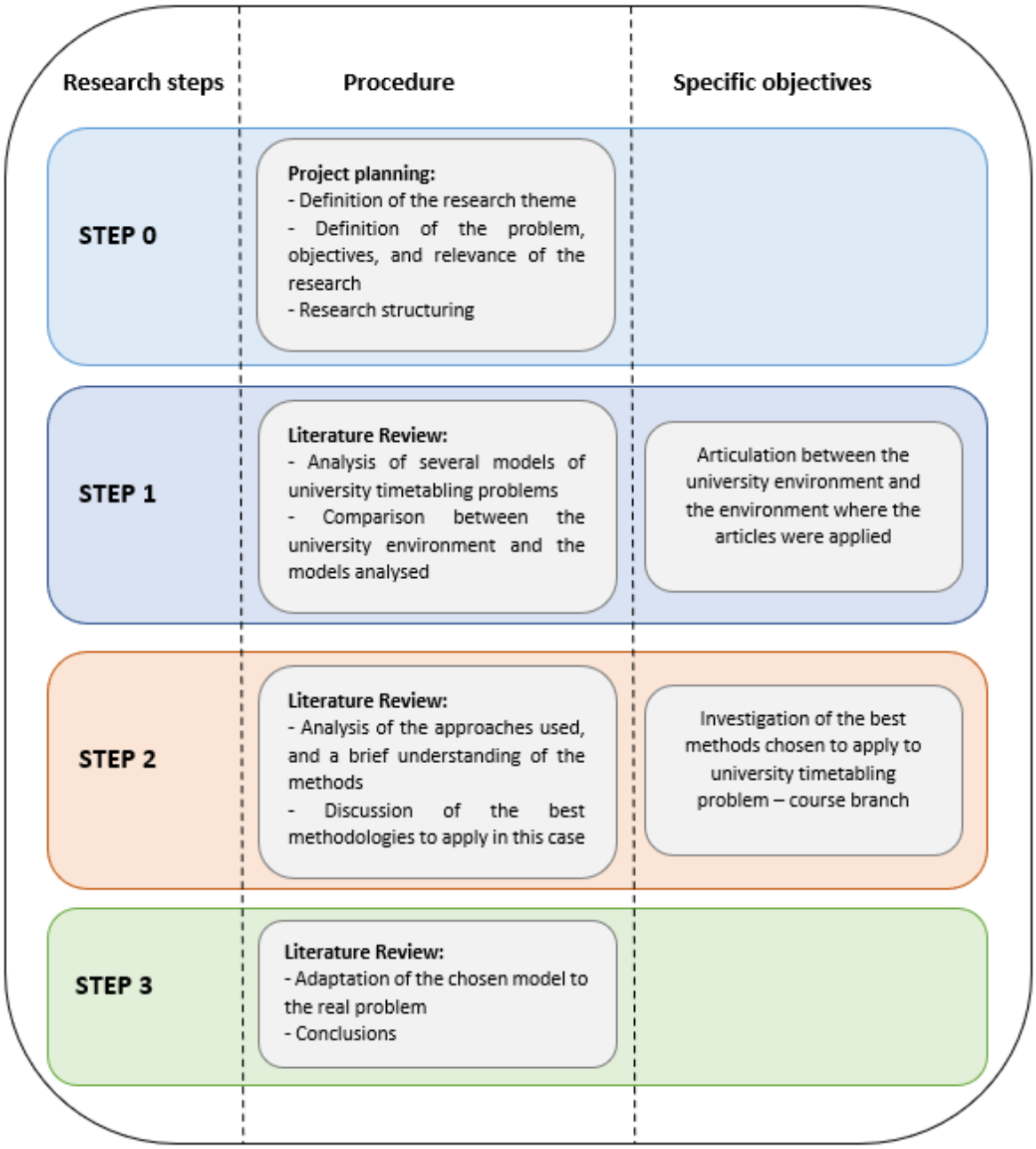


Figure 1 - Overview of the research steps, procedures and objectives

In concordance with Mehrad & Tahriri Zangehen (2019), the concept of qualitative research design stems from the sociology and anthropology revolutions. It has been defined as an investigation that aims to reveal the hidden meanings of various cultural and social phenomena. Some of the terms used to describe this field include cultural investigations, natural inquiry, phenomenological, post-structuralism, and postmodernism. On the other hand, the goal of a quantitative research design is to regulate the connotation of an independent variable or a consequence variable in relation to a population. This type of research can be described as experimental or descriptive.

Considering the two branches that compose the methodology, the present work is based on quantitative research due to the nature of the problem (timetabling in education) and the perspective for future solutions (algorithmic model and mathematical approaches).

Regarding the educational problem, there are seven different timetabling methods that can be used (Johnes, 2015). These include clustering-based methods, sequential methods, constraint-based methods, meta-heuristic methods, multi-criteria approaches, hyper-heuristics, and case-based reasoning. The examples present in the literature show the various kinds of problems that can be solved using timetabling techniques. Some of these include complex neighbourhood searches, multi-objective relaxations, and bipartite matches (Kingston, 2014).
Bashab et al. (2020) made a systematic review of this topic, to understand better the approaches used to solve this kind of problem. The authors started by mapping the articles published from 2009 until 2020 and the result consisted of 131 publications. The outcomes showed that the approaches were based on "hybrid methods (32%), in which the distribution of meta-heuristic algorithms the hybrid algorithms represent the higher application (31%)" and two-thirds of the searches developed were solution applications. This systematic review proves that university timetabling problems are often solved by resorting to meta-heuristic algorithms and "a new trend of meta-heuristic algorithms such as grey wolf optimizer, cat swarm optimization algorithm, Elitist self-adaptive step-size search and others with high expectations for reliable and satisfying results can be proposed to fill this gap".

Besides heuristic methods, there are some other methods pointed out in other surveys, such as graph colouring algorithm, local search (LS), tabu search, genetic algorithms (GA), constraints-based methods, and linear programming approach. Other approaches used and mentioned in the literature are artificial ant colony (ACO), hybrid bee colony optimization method (HBC), particle swarm optimization (PSO), neural networks (Dimopoulou & Miliotis, 2001; Firdaus Khair et al., 2018; Pandey & Sharma, 2016).

Chen et al. (2021) refer that there are many areas that find useful the development of timetabling research beyond education, for example: "transportation, hospitals, private enterprises, sports, and many others". There are multiple approaches used, highlighting meta-heuristics, hybrid methodologies, and hyper-heuristic that come up with real results to solve this combinatorial optimization problem.

This introduction outlines the context and relevance of the university timetabling problem and sets the stage for the project. It was intended the develop of an innovative algorithm that will optimize schedule construction, and also adapt to future needs and constraints, thus reducing the human resources and time dispended. Right through this article, the challenges faced at UCTP are explored along with the presentation of the methodological approach to solving this complex and critical problem.

1.4. Structure of the dissertation

This report is divided into chapters. In addition to the Introduction, this work contains the chapters of Literature Review, Methods and Applications, Results and Discussion and Conclusion.

The first chapter provides a brief introduction to the work carried out, covering the contextualization, methodological options, objectives of the project, as well as its organization.

In chapter two, a literature review is introduced, taking into account the themes covered throughout the project. Therefore, the theme of the timetabling problem is exposed, presenting its

division and focusing on problems related to the university course timetabling problem. The model chosen to adapt to the case study is also presented.

The third chapter, Methods and Applications, aims to describe the methods used: the current problem, the adapted mathematical model (parameters, variables, constraints, objective function), the structure of the data used to validate the model, and the structure of the computational application.

Chapter four presents the results obtained and a discussion about them.

In chapter five, a conclusion is made regarding the entire preparation of the project and some limitations found, as well as future research considered relevant.

Next, bibliographical references are presented, as well as annexes that contribute to a better analysis of the project.

2. LITERATURE REVIEW

This chapter presents a literature review aiming to frame the developed work. First, some of the concepts approached are explained, as well as the timetabling problem.

2.1. Educational timetabling

According to Aziz & Aizam (2018), a timetable can be described as "all the data one has to know regarding specific events that are scheduled to occur". Although it might seem a simple problem, reaching a satisfying solution can be difficult, especially due to all the parameters, variables, and limitations that must be considered. (Tan et al., 2021) point out that "the evolution of the educational systems are continuous, new challenges often arise", which implies adapting the methodologies and approaches considered. Although there are new developments emerging, it is not possible to "compare studies or rigorous analysis of these methodologies", due to the lack of existence of these works.

Educational timetabling problems can be divided into three significant types, in concordance with the three purposes associated with the development of the timetable: school timetabling, course timetabling, and exam timetabling problems. (McCollum et al., 2010; Tan et al., 2021)

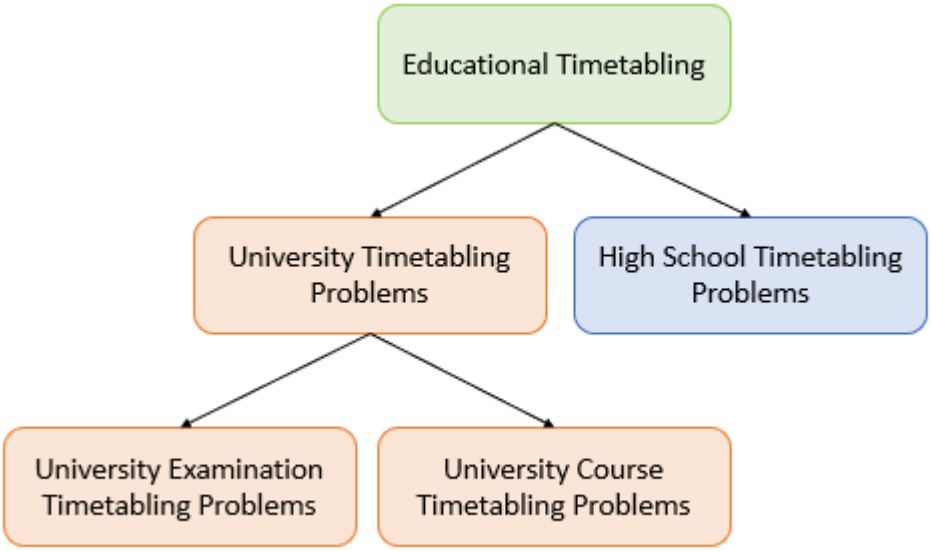


Figure 2 - Types of timetabling problems

2.1.1. School timetabling problem

The school timetabling problem is the one that appears to be less attractive to researchers since it hasn't been as developed as the other university timetabling problems. Considering the fewer variables and, most of the time, the smallest dimension of the problem, this area of educational timetabling is considered the simplest one. (Birbas et al., 2009; Pillay, 2016; Tan et al., 2021)

The school timetabling problem is combinatorial and complex, using several binary variables. Solving this type of problem depends on the dimension and other characteristics, which may vary

from school to school. Several approaches are referred to in the literature, such as algorithmic methods, or starting by assigning time slots to teachers and the solve the timetabling problem itself. (Sørensen & Dahms, 2014; Valouxis & Housos, 2003)

2.1.2. Examination timetabling problem in universities

On the other hand, examination timetabling problems are the ones more covered by the literature. This type of problem aims to specify for each period a particular exam, developing timetables with quality that mandatorily satisfy hard constraints. (McCollum et al., 2010) To solve the examination timetabling problem, the main solution goes by allocating exams while avoiding conflicts or overlays, which means, according to Carter & Laporte (1996), "assign examinations to a limited number of available time periods", eliminating the need for students to take multiple exams at the same time (Mirhassani & Habibi, 2013).

2.1.3. University course timetabling problem (UCTP)

University Course Timetabling Problem (UCTP) is the branch of educational timetabling that aims to assign students and faculty to the respective degree courses, classes, classrooms, and so on, according to the time slots and periods defined. (Aziz & Aizam, 2018; Carter & Laporte, 1998)

The UCTP is subdivided into two: curriculum-based and post-enrolment. While the timetable for the curriculum-based problem is developed without knowing the student enrolment at the time (enrolment data defined), the timetable for the post-enrolment problem is created in accordance with the student enrolment (the number of events is fixed). (Ceschia et al., 2012; McCollum et al., 2010)

2.2. State-of-the-art

In this topic, it is presented the approaches of the models and techniques used to develop the model of integer programming to solve the timetabling problem at *Instituto Superior de Engenharia do Porto*. The projects developed are briefly described, mentioning, when possible, the function objective, approach(es) used, parameters, variables, CPU time, solutions obtained, and other possible relevant information.

Firstly, considering the high school timetabling problems, a scientific review has been made by (Tan et al., 2021), approaching many countries' methodologies, aiming to understand which ones are used the most. This kind of problem is usually defined as NP-hard and, through the years, meta-heuristics and mathematical models are being more and more used. Timetabling problems are mainly composed of four different finite sets: resources (R), meeting (M), times (T), and constraints (C). The datasets analysed were prevenient from Australia, Brazil, Czech Republic, Denmark, Spain, Finland, Greece, Italy, Kosovo, Netherlands, England, USA, and South Africa. These showed that "twenty-four meta-heuristics, eight mathematical optimization methods, four hyper-heuristics and three matheuristics (the combination of meta-heuristics and mathematical optimization technique)" were used as approaches to solve the timetabling problem. As a conclusion of this scientific review, it is possible to understand that many software packages were improved, to

respect the constraints associated with this kind of problem, aiming to combine both the interface for the user and the optimization techniques.

In a Greek high school, the main issue is when teachers have to teach in several classes at the same time, and the students are still in their classrooms. To avoid violating any constraints, find a schedule that accommodates all the class sections. Valouxis & Housos (2003) define as objective the minimization of the amount of time that teachers spend idle while also trying to accommodate their requests for late or early shift assignments. This problem used a constraint programming (CP) approach to solve the timetabling issue for a typical high school. The techniques used in this study included local search and operations research (OR) models, to decrease the solution search space. the algorithm presented is based on ILOG Solver and C/C++. It aims to provide a safe and efficient method for performing complex computations. The CP engine's searching process is performed through a combination of computational steps. Experiments conducted on the project revealed that the CP engine was able to achieve better results than a standard search engine that only includes a couple of standard library functions. The total execution time (overall stopping rule) of the algorithm is computed by HP 9000=715=100 MHz workstations and was defined as 1 hour. The results of the tests revealed that the algorithm was able to find the optimal solution for the first and second problems. The average execution time for these problems is around 15 to 20 minutes. The other two problems were also very satisfactory. On the other hand, the total idle hours for the three problems were zero. This means that the maximum number of hours that the computer can use was utilized.

In the Western Greece Region, there are 23 teachers working in primary and secondary education. They are responsible for 16 full-time positions and seven part-time positions. Birbas et al. (2009) present a paper with a two-stage approach to solving the timetabling problem in schools. The first stage involves developing a shift assignment system that considers the preferences of the teachers. The second stage involves developing a school policy that addresses the needs of the curriculum and the core courses. The second stage of the problem was solved by implementing penalty functions on the coefficients in the timetabling problem objective function. The MIP solver of CPLEX 10.1 was used for the two problems. Several high schools have successfully used the approach. The size of the institution affects the teachers' preferences, as well as the schedule compactness, but it does not affect the index for core courses.

It is widely acknowledged that IP models are difficult to solve, and timetabling is mainly dominated by heuristic approaches, studied by Sørensen & Dahms (2014) in this article. The methodologies of an IP model were analysed and solved in a two-stage process for a case of a Danish school timetabling system. The first stage focuses on the design and implementation of the model, while the second stage involves the evaluation of the results (uses stage I as input). The goal of this approach is to use Hall's theorem to show how matchings can be found in a bipartite graph. In the basic form, this theorem yields an exponential number of constraints in Stage I.

For the implementation of Gurobi, it was used on a machine that had an Intel Core i7, 930 at 2.80 GHz, and 12 GB of ram. Windows 8 64bit was used. The time limit for running the program was set at 7200 s. To evaluate its decomposition, a database of high school students was used. For 20 instances, Gurobi was not able to resolve the issue with the extension of the Stage I Model's root node. As a result, it is not feasible to use the extended model to get good solutions.

The following table sums up these articles related to high school timetables.

Table 1 - Summary of the articles studied regarding the school timetable problem.

Authors	Methods & Approaches	Results
Tan et al., 2021	Scientific Review	Scientific review
Valouxis & Housos, 2003	Local search (Heuristic)	Satisfying (optimal solution)
Birbas et al., 2009	Two-stage IP	Very satisfying
Sørensen & Dahms, 2014	Two-stage IP	Unable to solve (stopping criteria)

The first university model that was analysed was a case study related to Engineering Schools of Greek Universities, namely the Electrical and Computer Engineering Department (Daskalaki et al., 2004). The binary integer programming (IP) formulation had as an objective function the minimization of costs, divided by two terms. The model developed was built based on:

- 6 distinct parameters (days of the week (I), time period of a day (J), group of students (K), professor (L), courses (M) and classrooms (N)).
- two different sets of binary variables (the basic set – that takes the value of 1 when course m, taught by professor l, to the group of students k, is scheduled for the jth period of day i, in classroom n; and the auxiliar variables that are related to the sessions – whether if they are multiple or not).
- constraints (uniqueness, completeness, consecutiveness, repetitiveness, pre-assignment constraints).

The mixed integer programming (MIP) solver by CPLEX 5.1 was used, performed by an HP J7000Workstation and a solution was obtained. To obtain the optimal timetables, the CPU time varied between 2.5 and 95 minutes, relative to the smallest and largest problem, respectively. The timetables constructed respected all the hard constraints imposed and satisfied most of the soft constraints, which was well acceptable to both faculty members and students.

According to (Schimmelpfeng & Helber, 2007), it is not widely clear if most of the proposed solutions for timetabling problems are effectively implemented. The referred authors applicated a timetabling model using integer programming, at Hannover University (Germany), namely the School of Economics and Management (150 different lectures per week, taught by around 100 professors). To solve this problem, it was mainly used the open-source mixed integer solver and CPLEX (on a 3 GHz Intel Pentium 4 processor with 4 GB RAM). The formulation developed considered essentially 4 parameters: time (A), rooms (B), courses and teaching groups (C), and teachers' preferences (D). The objective function, contrarily to most of the models created, does not consider costs but the minimization of violations (penalties) related to the preferences of the teachers (weekdays and rooms, for example). The schedules obtained were evaluated by the teachers and assistants, that were, in the great majority, satisfied: "In only 28 out of 181 cases, a teaching group was scheduled during a time slot that was not the teachers' priority. In 25 out of these 28 cases, a time slot with a second or third priority (out of a total of five) was assigned".

The university timetabling problem encountered by the IP models is a difficult problem to solve, due to the algorithm used for solving it being very complex. Prabodanie (2017) proposed a model to find a solution to this problem. The model was solved using the Open Solver add-in in Excel. A

binary integer program model was proposed to improve the model's efficiency and reduce the size of the problems. The decision variables were designed with fewer dimensions to fit the model's size. The results of the model indicated that timetabling models with computerized capabilities can produce more efficient schedules. The use of tools such as Open Solver and MS Excel was very productive in solving and modelling the timetabling problem. The programming capabilities of Excel allowed the creation of custom sets of variables. The model generated compact and efficient schedules. The time it took to calculate the schedules varied depending on the problem size (from a few seconds to 30 minutes).

The goal of the following project, presented by Mokhtari et al. (2021), was to develop a multi-objective programming model that would allow the management of postgraduate programs in Industrial Engineering at IAUN's Najafabad Branch. It was able to minimize the violations of the educational priorities and the students' travel time. The proposed model was then validated using the CPLEX software on a laptop with 4 GB of RAM and Intel Core i5, 2.5 GHz. The proposed model was able to perform three functions: minimize the violations of educational priorities, reduce students' travel time, and minimize the surplus capacity of classrooms. The study also investigated the constraints caused by classrooms, time slots, teachers' expertise, and days. The results of the study validated the proposed model's practicality and applicability, and it was able to provide a case study timetable that reflected the preferences and limitations of the case study. One of the exciting directions for research is the development of software that can schedule lectures. This should involve developing metaheuristic or heuristic methods that can handle large-scale timetabling issues.

Palma & Bornhardt (2020) propose a curriculum-based method for scheduling courses before the students register. It considers the various practical aspects when it comes to managing conflicts between lectures. A multi-objective programming model that takes into account the various factors that affect the scheduling of courses was developed. It is able to maximize the number of time slots allocated for each course and provide flexibility for students who are stragglers. The model was created using the commercial software known as Lingo, which was able to be used in combination with MS Excel. It was then tested and compared with the previous solution. The first semester of classes for the 2017 academic year was scheduled for the faculty of engineering at the University of Chile. The results of the tests revealed that the model performed better than the actual solution, which was created by two individuals over a period of two weeks. The most challenging part of the program was handling the thousands of decision variables and 214,000 constraints. It took about an hour to solve the problem, and a 5% gap was automatically defined. It also successfully managed various practical issues that were not addressed in the previous solution. One of the main reasons why the model performed better than the manual method was due to the increasing symmetry of the schedule.

The following table sums up these articles related to the UCTP, regarding binary or mixed integer programming formulations.

Table 2 - Summary of the articles related to University Course Timetabling Problem, using binary or mixed integer programming.

Authors	Methods & Approaches	Results
Daskalaki et al., 2004	Binary IP	Satisfying (OS found)
Schimmelpfeng & Helber, 2007	Mixed IP	Satisfying
Prabodanie, 2017	Binary IP	Satisfying (compact TT)
Mokhtari et al. 2021	Mixed IP	Satisfying
Palma & Bornhardt 2020	Mixed IP	Better solutions

Another paper presented by Rudová & Murray (2002) describes the process involved in developing a timetabling system for Purdue University. It shows how the system will be used to manage the course load of almost 30,000 students. It describes and applies a logic programming extension to the constraints that are used in the timetabling system. It allows for the partial satisfaction of the soft constraints. It was used a weighted CSP (constraint satisfaction problem) approach to minimize the overall dissatisfaction with the constraints, which considers the various weights and costs associated with each constraint. Currently, the system handles three kinds of soft constraints: two types of unary constraints related to faculty preferences, and binary constraints. A branch-and-bound search was performed to improve the system's cost function. The solution improved through four repair searches, but additional steps failed to improve its quality. The system was able to meet 94% of the students' demands and 90% of the teachers' preferential requirements. Unfortunately, it encountered issues with the implementation of the ILOG Scheduler's bound consistency algorithms. Most of the time, classes are taught at times when there are not enough available rooms.

It is important that the various administrative tasks related to these activities are carried out according to the rules. The goal of the project presented by Akif Bakır & Aksop (2008) is to develop a programming model that minimizes the discomfort experienced by students and lecturers while at the same token implementing rules that are bound by constraints. This model can be used to implement new criteria or rules. This problem was solved using the software program, Lingo 8.0, which was installed on a computer platform consisting of a 256 MB Ram, a 2.6 MHz, and a Pentium 4. The algorithm used for the problem was a branch and bound method, which was able to minimize the total dissatisfaction factor while still meeting the model's constraints. Although it took about five minutes to come up with a feasible solution, it took a long period of time to get an exact optimal result due to the limitations of the computer platform. Due to the complexity of the problem, it is difficult to find an optimal solution (OS) that can reduce CPU time. Thus, further research is needed to develop a strategy that can use heuristics.

Table 3 sums up the articles that describe UCTP, using branch and bound method.

Table 3 - Summary of the articles related to University Course Timetabling Problem, using branch and bound method.

Authors	Methods & Approaches	Results
Rudová & Murray, 2002	Branch and bound	Needs improvement
Akif Bakır & Aksop, 2008	Branch and bound	Satisfying (OS not found)

Due to the complexity of operational rules and the computational challenges presented by UCTP, many researchers have abandoned their efforts in modelling these problems. Scheduling problems are an issue that university personnel encounters several times throughout the academic year. They have to make various decisions regarding the assignments of students, faculty members, and course schedules. (Daskalaki & Birbas, 2005) The paper developed by the authors mentioned above presents a two-stage relaxation method that effectively addresses the timetabling problems of universities. The first stage of the relaxation process is performed on the constraints that apply to multi-period courses. These constraints are usually harder to recover due to their computational complexity. The second stage involves solving the various subproblems that are related to the schedule, for a locally optimal solution. To solve the mixed-integer programming (MIP) problem, CPLEX 5.1 was used in an HP J7000 workstation. The computational time for various tasks was reduced to 2.5 minutes for 25 courses and 96 minutes for 92 courses. When the computation time is not an issue, new requirements can be added to the schedule to make it more user-friendly.

Large student flows between classes in university buildings can cause congestion issues. These problems can lead to long queues at elevators and stairwells, which can delay the start of lectures. The conceptual model presented by Vermuyten et al. (2016) is used to analyse the data collected from the KU Leuven Campus in Belgium. It is validated and tested with 21 case studies. The goal of this paper is to develop a course timetable that is designed to minimize the flow of students through the university. The first stage involves assigning lectures to rooms and timeslots, while the second stage transfers classrooms to the timetable. The model is created using Microsoft Visual Studio 2013's CPLEX 12.3 library. It is executed on a computer with an Intel Core I7-6400U processor and 8GB of RAM. Through a comprehensive computational analysis, we were able to show that our two-stage approach to designing software for UCTPs can provide high-quality solutions while reducing student flows. The first stage of this method aims to provide a timetable that is both feasible and flexible with regard to the hard constraints, while the second stage takes into account the input. It can be used for various types of education.

In another study, conducted by Rappos et al. (2022), the goal of this algorithm is to provide a feasible solution to the problem of class time assignment. It involves a first-stage optimization procedure and a second-step local search to improve the solution's value. The four decision variables used in this formulation are: x, z, y, and Z. These variables are used to represent the various aspects of the assignment, such as the classroom assignment and the allocation of classes. The objective of the problem is to minimize the time allocated for the assignment and the various constraints that it involves, such as the distribution of classes and classroom allocation. This is done through a combination of a mixed-integer and a linear programming model. The program was developed using Java, CPLEX, and Gurobi, and virtual machines with 32GB RAM and four cores were created. After a feasible solution was obtained, an improvement heuristic iteratively is carried out to increase the quality of the solution. This method was very effective for small and medium-sized

problems but not for large ones. In most cases, the first stage of the process is completed in about 1 hour, and in almost 70% of the cases, less than 2 hours.

Sylejmani et al. (2022) present the solution for the new variation of the university timetabling problem that was introduced in the fourth international timetabling competition: a two-phase approach that involves finding a feasible solution and optimizing it. The first phase is focused on finding a solution that is feasible, while the second phase involves penalizing the solutions to force them into new neighbourhoods. The solvers were able to detect local optimal and perform a specialized search for constraints that are difficult to satisfy. The computed worst-case scenarios are used to evaluate the solutions. These include the worst room penalty, the worst student penalty, and the worst distribution penalty. The simulated annealing algorithm considers the changes in the solution's condition when it is modified. The experiments were performed on a machine with a 1x Server System Building block PROnex 1029.GQ-TRT. It was equipped with 2x Intel Xeon E6130 2.10GHz 16MB Cache, 2x 2TB solid-state drives, and 12x 8 GB DDR4-2666MHz ECC memory. The results of the tests revealed that the solutions had a gap of less than 15 percent from the best-known solution in five out of 30 instances. In addition, the average gap between the best and the most common solutions in the middle and late chunks of the instances is around 80%, 90%, and 230%, respectively. As part of future work in the field of timetabling, the authors propose developing additional search operators that are focused on finding solutions that are feasible and reducing the overall complexity of the problem.

The initial promise of metaheuristics and heuristics was eventually broken down as attempts to solve large timetabling problems proved costly. This issue is attributed to the lack of exploitation of the underlying structure and the fragmentation of the solution and modelling approaches (Siddiqui & Arshad Raza, 2021). This article presents a timetabling ontology that aims to provide a common modelling framework for addressing various real-time applications. It can be used to map the anatomy of timetabling problems to its general structure. The goal of this approach is to provide a two-stage solution for addressing the generalized problem of timetabling. The first stage involves generating elite solutions that are based on the general problem structure. The second stage, which involves using a metaheuristic, can be used to improve the solutions' performance. The framework was developed using a Python 3.7.6 algorithm, which was executed on a Windows-based computer with an 8-core Intel i7-8565U processor. It was easy to notice that the computational time and quality of the solution are affected by the scarcity of resources. To demonstrate the effectiveness of the methodology, it was conducted a timetabling problem that was designed to test its performance. The results of the study revealed that the initial solutions were generally close to their lower bound, even for large problems.

Table 4 outlines the articles that describe two-stage approaches, regarding UCTP.

Table 4 - Summary of the articles related to University Course Timetabling Problem, using a two-stage approach.

Authors	Methods & Approaches	Results
Daskalaki & Birbas, 2005	Two-stage relaxation method	Satisfying (low CPU time)
Vermuyten et al., 2016	Two-stage IP	Feasible results (stage I)
Rappos et al. 2022	Two-stage (heuristics and LS 2nd)	Very effective for medium size problems
Sylejmani et al. 2022	Two-stage approach	Satisfying solutions
Siddiqui & Arshad Raza, 2021	Two-stage (metaheuristics 2nd)	Satisfying (relying on MH)

One of the most innovative methods for solving the UCTP issue is by implementing a hybrid algorithm, which allows the designer to automate the timetabling process. Rezaeipanah et al. (2021) present a hybrid algorithm that combines the advantages of the Improved Parallel Genetic Algorithm and Local Search (IPGALS) and the Local Search (LS) approach to solve the course timetable problem. The LS method is used to strengthen the Genetic Algorithm (GA), while the IPGALS has been designed to ensure that the hard constraints are never violated. The performance of the algorithm is evaluated using the Distance to Feasibility (DF) criterion, which helps to promote the practicality of the solution and measures the hard constraints. It helps in developing effective solutions and promoting the performance of the algorithm. The experiments were performed on an Intel Core i5, 8 GB of memory, and Windows 10 64-bit, using MATLAB for the implementation. The results of the experiments show that the proposed algorithm is more effective than the other methods that are commonly used to solve the UCTP issue. The GA is an efficient method for the UCTP since it has a multi-directional search. However, it might get stuck in the local optimal trap due to its focus on extraction. This paper presents a couple of methods that help prevent this issue from happening and points to that as future research.

A paper presented by Méndez-Díaz et al. (2016) depicts a timetabling problem that is related to a private university's case study in Argentina. A framework that combines an ILP model with a heuristic approach to solve this problem was developed and it can be used successfully in practice and produce good quality solutions. The experiments were performed on a workstation using an Intel Core i7 CPU and 16 GB of RAM. The code for the algorithms was developed in C++ using the Callable Library in CPLEX. The results indicate that considering the various priorities has a significant impact on the solution's overall satisfaction. It also shows that allowing students to provide their own preference list has a positive effect on the overall solution's selection and the overall performance, despite the increase in computational times. The first feasible solution can be obtained after about 30 minutes of work. On average, around 10 solutions are generated after 3h. The entire algorithm can be run in about 10h. The timetabling process has been significantly reduced by implementing the new method, which allows the scheduler to create a timetable in just a few days, instead of two months. It allows them to select the most appropriate timetable according to the needs of the users. The implementation of the new method has also allowed students to make their own program and semester selection.

The next paper, written by Dimopoulou & Miliotis (2001), presents the design and development of a computer system that is built on an integer programming model, which is used to assign courses to time slots and rooms. It is also equipped with a front-end device, which allows writers to present the schedule in a more accurate manner. The objective function of this system is to use the knowledge and experience of the user to develop a better course-timetabling solution. The first solution is then built on a framework that considers the various factors that affect the schedule. A heuristic algorithm is then developed to improve the system's performance. The formulation of this framework is based on the concepts related to the grouping of courses and time periods. The system is designed to allow the user to enter, modify, and display all the necessary data. The system was developed using a PC-type computer with Microsoft Access 97. The IP model was then solved using the MPCODE package and XPRESS-MP, and the data from the University of Athens was used for the analysis. The results of the tests revealed that the system was able to find the optimal solution to a particular problem in less than a minute.

Burke & Petrovic (2002) aim to introduce the reader to some of the recent developments in timetabling problems being studied by the Automated Scheduling, Optimization and Planning Research Group (ASAP) at Nottingham. The paper presents various approaches to solving timetabling problems. The first part focuses on evolutionary and heuristic timetabling algorithms. The paper then describes two new methods for dealing with large-scale timetabling problems. And lastly, presents a multi-criteria decision problem framework that considers timetabling issues as a type of problem. It also discusses a case-based reasoning (CBR) method that can be used to solve such problems. The results of several experiments show that the CBR approach is more effective at solving timetabling problems with respect to their cost function than the graph heuristic method. But the graph heuristic method is more efficient when there are not enough cases to adapt to.

The complexity of university timetabling is a major issue that all universities must tackle in practice. The goal of this problem, presented by Mikkelsen & Holm (2022) is to develop a semester timetable that provides a room and time for all course events. It is feasible if the schedule meets all of the hard constraints, and the soft constraints are satisfied. The goal of the International Timetabling Competition (ITC) is to develop a parallelized mathematical model that can produce initial solutions and improve them later. This method uses the MIP model's full scope to calculate lower bounds and is combined with another method. Intel Core i7 dual-core processors and 768 GB of RAM were used and Gurobi 9.0, which is a MIP solver, on four threads. Results of the computational analysis show that the proposed matheuristic is a good choice for advancing the search. It can find optimal solutions for every competition scenario, and it performed well in the ITC 2019. The CPU time varied from a few seconds to approximately 75 hours. The proposed matheuristic can be further improved by developing a diversification scheme that takes into account the different constraints.

The next table reviews the articles related to heuristics and combinations with heuristics for the university course timetabling problem.

Table 5 - Summary of the articles related to University Course Timetabling Problem, using heuristics.

Authors	Methods & Approaches	Results
Rezaeipanah et al., 2021	IPGALS	Problems local optimal trap
Méndez-Dias et al. 2016	ILP + Heuristic	10 solutions found
Dimopoulou & Miliotis 2001	IP + Heuristic	Optimal solution found
Burke & Petrovic, 2002	Graph heuristic method	More efficient than before
Mikkelsen & Holm, 2022	Parallelized Matheuristic	Satisfying solutions (some OS)

The complexity of a university's timetabling problem typically involves scheduling thousands of courses for hundreds of instructors over several hundred classrooms. To solve this problem, Sarin et al. (2010) present a solution to this problem based on the partitioning of the integer programming model created by Benders. This method considers the special properties of the underlying formulation. The paper presents a comprehensive solution to the timetabling problem of Virginia Tech, which involves scheduling meetings between the students and the faculty members. It effectively addresses the various constraints of the university's resources and accommodates the preferences of the faculty members while reducing the distance that the faculty members travel to the classrooms. CPLEX 9 was used to implement the formulation code in AMPL and to run the computation on a Dell Precision 340. The computer has 2 GB of RAM and a 2.8GHz CPU. The results of the study revealed that the methodology used for calculating the course schedules significantly improved the quality of the results. The CPU maximum required to complete the task was only 90 minutes.

Song et al. (2018) presented a paper that proposed an iterated version of the course timetabling problem that is focused on finding a feasible solution. The framework for this algorithm is composed of three phases: intensification, diversification, and initialization. The evaluation of the proposed algorithm is performed against seven existing methods in the literature, which is not commonly seen in the literature. The proposed algorithm was developed using C++, and it was carried out on a personal computer using a 2.06 GHz Intel Core 2 Quad CPU and 1.0 GB of memory. Some of the other references that were used included the EIS, SA-M, and HAD algorithms, which were run on a different platform using a 1.6 GHz i7 CPU. Out of 60 instances, the computational results of the proposed algorithm generated 58 feasible solutions. For the two large cases, the upper bounds were increased.

Siddiqui et al. (2018) present a multi-stage problem that involves academic term preparation. The system is designed to automate the spreadsheet process for a large business school in Saudi Arabia. The system is designed to automate the various stages of the workflow by providing a web-based application that handles all the tasks involved in the preparation of the term sheet. It also features an optimization module that can be used to improve the efficiency of the process. The solution was performed using the ATPS framework, which is a timetabling engine that uses Python. The model was generated using a computer running Windows 7 using a 64-bit Intel i7 processor and 64 GB RAM. The Gurobi software, which is a Python interface, was used to perform the optimization. A semi-automatic spreadsheet tool was replaced by the new system, which significantly improved

the quality of the timetable and reduced lead times. The various stakeholder groups have indicated that the system has provided them with several benefits. Some of these include its ability to improve the quality of the timetable and reduce the lead times. It also provides a web-based application that allows them to manage their timetabling activities. In addition, it has the capability to eliminate human errors in the solution.

Aziz & Aizam (2018) collaborated with a university in Malaysia to develop a model that can be used to solve the timetabling problem. They tested it using experimental data. The team utilized CPLEX and AIMMS software to find the most effective solution. Various features that can be used in the timetabling process to improve the satisfaction of students and teachers were found. These features could potentially change the way the teaching and learning environment is conducted. Through the review of the literature, the team was able to produce a general mathematical model that can be utilized to develop a more effective solution for the timetabling issue, and eventually seven new constraints were added to the model. These new constraints were designed to address the requirements of the timetabling community. The goal of this article is to review the literature on the topic of timetabling. In the future, the authors say they will continue to develop a mathematical model that will be used to develop a more robust solution for the course timetabling issue.

The table below sums up the articles described above.

Table 6 - Summary of the articles related to University Course Timetabling Problem (several approaches)

Authors	Methods & Approach	Results
Sarin et al., 2010	Benders' partitioning	Improvement
Song et al. 2018	3 phases	General feasible solutions
Siddiqui et al. 2018	Web-based application	Feasible results
Aziz & Aizam, 2018	Scientific Review	Scientific review

2.3. Discussion and critical analysis

Analysing all the models cited above, it is possible to conclude that most of the models present essentially five parameters: periods of time (at what time lessons will be taught), classrooms (where the lessons will be taught), groups of students (who is attending the lessons), faculty (who is teaching or responsible for the classes), and course units (which lessons are being taught).

The variables used depend essentially on the type of approach used to solve the problem, but the most frequent were binary variables that define if a course is associated to a faculty member, group of students, period of the day and also to a classroom, simultaneously.

Regarding the objective function of the models, they vary from the objective of each educational institution, such as:

- Minimize the time allocated for the assignment and the several constraints that it involves.
- Minimize the violations of educational priorities/ overall dissatisfaction with the constraints.
- Minimize the excess capacity of classrooms.

- Minimize the cost and resources used.
- Minimize the time that faculty members spend idle.
- Minimize the travel distance between rooms (regarding students and/or professors).

It is possible to conclude through the studies analysed that high school timetabling problems are solved as smaller problems than the university ones. The methodology that is frequently used to solve these problems is a two-stage approach, obtaining satisfying solutions within minutes. Usually, also for the UCTP, the first phase aims to find a feasible solution, and the second phase is used to improve the solution previously obtained, which increases the computational time but can output better solutions.

On the other hand, to solve the University Course Timetabling Problem, many approaches are tested. For example, the Branch and Bound method was presented twice above, and the results are not satisfying in 50% of the cases. The rest of them, suggest improvements to reach the optimal solution. The two-stage approaches are very applied, but the second phase can vary from local search, heuristics, metaheuristics, relaxation methods, integer programming, and more. All of them present feasible results concluding stage I. The CPU varies from minutes to a couple of hours, depending on the size of the problem: the bigger it is, the longer it takes. The notorious improvements were noticed when using LS and the relaxation method in the second stage. Integer programming is frequently used and can come up with optimal solutions, and generally very satisfying results, in less than 2 hours. Heuristics, metaheuristics, and matheurisitcs were also present in the literature. The satisfying results could be obtained in a few seconds or almost 80 hours, but it was more frequent to find optimal solutions than in the other methodologies. The biggest disadvantage is the high cost associated with computational issues, which arises as the size of the problem increases. Other approaches were mentioned, such as Benders' partitioning (linear programming), case-based reasoning (adapting other solutions), IPGALS (combination between IP, genetic algorithms, and local search), and web-based application/computer systems developed. Some of these achieve feasible solutions, but in most of the cases, suggestions for improvements are highlighted.

The majority of the articles presented previously mention that the timetables' elaboration was mainly made by hand and that spent considerable resources (human, material, and essentially time). This way, a support system based on an algorithm can minimize the resources used. In this specific case, regarding ISEP, the preferable objective function would be to minimize the costs associated and the violations of the constraints (that will consider the faculty requests), following an integer programming approach to reach the first feasible solutions and, if needed, consider heuristic approaches to find optimal solutions, when possible.

Considering all the information above regarding course-timetabling development, an analysis was made to compare and search for an adaptable mathematical model. The model based on the article "An Integer Programming Formulation for a Case Study in University Timetabling" (Daskalaki et al., 2004) was chosen to be adapted to ISEP's reality. This model follows an integer programming formulation, sharing the same general features with DEM:

- I as day of the week.
- J as period of the day, on which classes might be scheduled, single classes not considering the repetition of periods.
- K as group of students, for which a course in the timetabling is designed.

- L as professor, lecturer or other teaching staff – faculty.
- M as courses to be scheduled for a given set of groups of students of each degree.
- N as classrooms available for scheduling courses for a given set of groups of students.

Based on the six sets above, many subsets were created to reduce the number of variables, keeping them controllable:

- K_l – group of students for which professor *l* offers some course.
- L_i – professor available on day *i.*
- L_k – professor teaching at least one course for the group of students *k*.
- L_m – professor teaching course *m*.
- L_{km} – intersection between L_k and L_m (faculty members that teach one course to at least one group of students).
- L_{ki} – intersection between L_k and L_i (faculty members available on one day to teach to a group of students).
- M_k – course designed for the kth group of students or offered by the kth division for its students.
- M_l – course taught by teacher *l*.
- M_n – course designed for a group of students that fits in classroom *n*.
- M_{kl} – intersection between M_k and M_l (courses taught by a faculty member *l* to a given group of students *k*).
- M_{kn} – intersection between M_k and M_n (courses taught to a given group of students *k* in a classroom *n*).
- M_{kln} – intersection between M_k, M_l and M_n (courses taught to a group of students *k* at a classroom *n,* by a professor *l*).
- M_{lab} – courses *m* that require lab work.
- N_{mk} – classroom *n* that fits the group of students *k* for the course *m*.
- I_n – day *i* on which classroom *n* is available for use.
- I_l – day *i* on which teacher *l* is available for teaching assignment.
- I_{ln} – intersection between I_l and I_n (days on which at least one professor *l* is available to teach at classroom *n*).
- J_{iln} – time period *j* of day *I* on which teacher *l* and classroom *n* are available for assignments.
- JL_{iln} – time interval from period j_a to j_b of day *i* on which teacher *l* and classroom *n* are available for assignments.
- P_m – used for multi-period classes (more than the usual 50minutes/1hour). $P_m = p_v \in \{1,2,\dots,pmax_m\}$, where $pmax_m$ is the number of repetitions of 1-hour sessions that a given course *m* requires.
- H_m – used for multi-period classes (more than the usual 50minutes/1hour).
- PRA – only to be used in case there are pre-assigned constraints.

The IP model works with two sets of decision binary variables, one as the main variable and the other as an auxiliary one, to help solve the repetitive classes:

- $x_{i,j,k,l,m,n}$ – takes the value of 1, when course *m*, taught by teacher *l* to the group of students *k*, is scheduled for the *j*th period of day *i* in classroom *n*.

– $y_{i,p_v,k,h_v,m,n}$ – auxiliary variable (*p* and *h* are natural numbers); refers to lecture sessions: when classes are non-repetitive, p_v takes only the value of 1.

The model presents several constraint sets. The first three constraints are uniqueness constraints, that guarantee the non-existence of conflicts.

– Equation (1) ensures that every member of the teaching staff shall be assigned at most one course, one group of students, and one classroom at a time:

$$\sum_{k \in K} \sum_{m \in M_{kl}} \sum_{n \in N_{mk}} x_{i,j,k,l,m,n} \leq 1, \forall i \in I, \forall j \in J, \forall l \in L_i \quad (1)$$

– Equation (2) guarantees that for every group of students at most one course *m*, one teaching person *l* and one classroom *n* shall be assigned to every teaching period *j*:

$$\sum_{l \in L_{ki}} \sum_{m \in M_{kl}} \sum_{n \in N_{mk}} x_{i,j,k,l,m,n} \leq 1, \forall k \in K, \forall i \in I, \forall j \in J \quad (2)$$

– Equation (3) assures that every classroom may be assigned to at most one course, one teacher, and one group of students at a time:

$$\sum_{k \in K} \sum_{l \in L_{ki}} \sum_{m \in M_{kln}} x_{i,j,k,l,m,n} \leq 1, \forall n \in N, \forall i \in I_n, \forall j \in J_{in} \quad (3)$$

Equations (4), (5), and (6) assure that the timetable is complete (completeness constraints).

– All courses in the curriculum of each student year should be in the timetable and in the right amount of teaching periods (a_k) – equation (4):

$$\sum_{l \in L_k} \sum_{m \in M_{kl}} \sum_{n \in N_{mk}} \sum_{i \in I_{ln}} \sum_{j \in J_{iln}} x_{i,j,k,l,m,n} = a_k, \forall k \in K \quad (4)$$

– Each course should be scheduled for as many teaching periods as the curriculum of each group of students requires (b_m) – equation (5):

$$\sum_{n \in N_{mk}} \sum_{i \in I_{ln}} \sum_{j \in J_{iln}} x_{i,j,k,l,m,n} = b_m, \forall k \in K, \forall l \in L_k, \forall m \in M_{kl} \quad (5)$$

– To ensure that each person in the teaching staff should be assigned to so many teaching periods as his/her weekly teaching load requires (s_l), follows equation (6):

$$\sum_{k \in K} \sum_{m \in M_{kl}} \sum_{n \in N_{mk}} \sum_{i \in I_{ln}} \sum_{j \in J_{iln}} x_{i,j,k,l,m,n} = s_l, \forall l \in L \quad (6)$$

The next set of constraints - equations (7), (8), and (9), are consecutiveness constraints, that ensure that the timetable may manage requests for multi-period sessions in some courses.

– A course *m* requiring a session of h_v consecutive periods should be assigned exactly h_v periods on a given day – equation (7):

$$\sum_{j \in JL_{iln}} x_{i,j,k,l,m,n} - \sum_{h_v \in H_m} \sum_{p_v \in P_m} (y_{i,p_v,k,h_v,m,n} * h_v) = 0\,, \quad \forall i \in I, \forall k \in K, \forall l \in L_k, \forall m \in M_{kl,} \forall n \in N_{mk} \tag{7}$$

– Equations (8) and (9) assure that if h_v periods of a given day have been assigned to course *m*, they should also be consecutive*:*

$$\forall i \in I, \forall k \in K, \forall l \in L_{ki}, \forall m \in M_{kl,} \forall n \in N_{mk}, \forall j_a \in FJL_{iln}, \forall h_v \in H_m \wedge h_v > 1, \forall t \in \{1, \dots, h_v - 1\}, x_{i,j_a,k,l,m,n} - x_{i,j_a+t,k,l,m,n} \leq 0, \tag{8}$$

$$\forall i \in I, \forall k \in K, \forall l \in L_{ki}, \forall m \in M_{kl,} \forall n \in N_{mk}, \forall j \in J_{iln}, \forall h_v \in H_m \wedge h_v > 1, \forall t \in \{2, \dots, h_v\}, -x_{i,j,k,l,m,n} + x_{i,j+1,k,l,m,n} - x_{i,j+t,k,l,m,n} \leq 0 \tag{9}$$

The repetitiveness constraints – (10), (11), (12), are also connected with the idea of consecutiveness in the sense that they secure the existence of the right amount of sessions of a certain type as well as the right amount of repetitions of a given session in case of repetitive recitations or lab work.

The h_v-period sessions should be as many as required during the week and for the non-repetitive parts of courses there should be at most one session per day. b_{m,h_v} is the total number of h_v-period sessions required for course *m* during a week.

– Equation (10) refers to courses that require non-repetitive sessions, lectures, or recitations delivered just once to their audience. On the other hand, equation (11) secures the existence of at most one of these sessions per day:

$$\forall k \in K, \forall m \in M_k - M_{lab}, \forall h_v \in H_m, \forall n \in N_{mk}, \sum_{i \in I_{ln}} \sum_{p_v \in P_m} y_{i,p_v,k,h_v,m,n} = b_{m,h_v} \tag{10}$$

$$\forall k \in K, \forall m \in M_k - M_{lab}, \forall l \in L_{km}, \forall h_v \in H_m, \forall n \in N_{mk}, \forall i \in I, \sum_{h_v \in H_m} \sum_{p_v \in P_m} y_{i,p_v,k,h_v,m,n} \leq 1 \tag{11}$$

– In case of repetitive parts of courses, like the lab work (PL) for a given course, the scheduling of the h_v-period sessions does not require different days for different sessions, since each session is delivered to a different sub-group of the *k*th group of students and the previous two constraints are replaced by Eq. (12). $pmax_m$ is the number of student sub-groups and therefore the number of repetitions required for the h_v-period session of course *m*:

$$\sum_{n \in N_{mk}} \sum_{i \in I_{ln}} \sum_{p_v \in P_m} y_{i,p_v,k,h_v,m,n} = pmax_m, \forall k \in K, \forall m \in M_{lab}, \forall h_v \in H_m \tag{12}$$

The last set of constraints (pre-assignment constraints) ensures that certain courses will be assigned at a given period in a given day and could be used either for the exact pre-allocation of courses or for the facilitation and better handling of computational difficulties.

- This set is composed of one constraint – equation (13), that guarantees that course *m* taught by teacher *l* to student group *k* should be assigned to a given period in a given day:

$$x_{i,j,k,l,m,n} = 1, \forall (i,j,k,l,m,n) \epsilon\ PRA \tag{13}$$

As can be seen in Eq. (14), the objective function aims to minimize a cost function consisting of two terms. The first term of the objective function refers to the cost of assigning course *m* to the *j*th period of the day *i*, while the second term refers to the cost incurred from the assignment of those courses that require sessions of more than one consecutive hour, on a given day of the week.

$$\begin{aligned} minimize \sum_{k \in K} \sum_{l \in L_k} \sum_{m \in M_{kl}} \sum_{n \in N_{mk}} \sum_{i \in I_{ln}} \sum_{j \in J_{iln}} c_{i,j,k,l,m,n} * x_{i,j,k,l,m,n} \\ + \sum_{k \in K} \sum_{i \in I} \sum_{m \in M_k} \sum_{n \in N_{mk}} \sum_{h \in H_m} \sum_{p_v \in P_m} a_{i,p_v,k,h_v,m,n} * y_{i,p_v,k,p_v,m,n} \end{aligned} \tag{14}$$

3. METHODS AND APPLICATIONS

On this chapter, it is presented the transition from the literature review to the practical scenario to adapt the chosen model to the case-study environment, starting with a brief context and description of DEM and its reality; followed by the presentation of the adapted model, as well as significant changes made. Besides this, the data used to perform the tests to validate the model are also described and the structure of the computational application (that combines Python, Pyomo and Microsoft Office Excel).

3.1. Problem Description

The present work aimed to implement a pilot project at DEM (Mechanical Engineering Department) at ISEP. This department has more than 1800 students, spread over 3 bachelor's degrees, 5 master's degrees, and 13 postgraduate degrees, with more than 100 members of the faculty.

The syllabus of each degree is organized by semesters and years. The groups of students are defined based on the maximum number of students by lecture type and each faculty member is assigned to the specific courses and classes.

At ISEP, timetables display information regarding the course, type of class classroom, faculty member, day of the week, and period of the day for each group of students. As the image below shows (figure 3), for the group of students 1DA (normally identified by 3 characters – one number regarding the degree's year, the first letter considering the type of group classes – day D or night N, and the last character to define the class), faculty members are represented by 3 letters (e.g. MMA), courses are represented by 5 characters (exceptionally 6), combining numbers and/or letters (e.g. IENG2, DESET-M), types of classes are defined according to theoretical (T), theoretical-practical (TP), practical-laboratory (PL), tutorial guidelines (OT) and seminars (S), classrooms are represented by the letter of the corresponding building, floor, and classroom number (e.g. F317, I201). Classes can start at 8 a.m. on Monday, and every weekday is available to schedule classes, until 8 p.m., for normal degrees.

Besides this information, each degree is defined mainly by 3 to 5 letters, that sum up the type of degree – the first one: L as BSc, M as MSc, and PG as postgraduate, and the following ones specifying the engineering area (e.g., LEM, MEGI, MEGCA).

Hora	Segunda-feira		Terça-feira		Quarta-feira		Quinta-feira		Sexta-feira	
08h10 08h30										
08h40 09h00			FISIC	PL	MATE1	TP	MECA1	TP	DESET-M	TP
09h10 09h30			F202 H503	JLL	F221	MPS	F208	HML	F209	MMA
09h40 10h00	DESET-M	PL								
10h10 10h30	F317	MMA	MTMET	T	IENG2	T	MATE1	T	MTMET	T
10h40 11h00			I201	AGM	I201	ANT	I201	JDM	I201	AGM
11h10 11h30			MATE1	T						
11h40 12h00	IENG2	PL	I201	JDM	MECA1	T	MTMET	TP		
12h10 12h30	F203	ANT	FISIC	T	I201	HML	F208	FID		
12h40 13h00			I201	APA						
13h10 13h30										
13h40 14h00										
14h10 14h30										
14h40 15h00										
15h10 15h30										
15h40 16h00										
16h10 16h30										
16h40 17h00										
17h10 17h30										
17h40 18h00										
18h10 18h30										
18h40 19h00										
19h10 19h30										
19h40 20h00										
20h10 20h30										
20h40 21h00										
21h10 21h30										
21h40 22h00										
22h10 22h30										
22h40 23h00										
23h10 23h30										

Figure 3 - Timetable example (https://www2.isep.ipp.pt/horarios/)

The creation of timetables at ISEP is a manual process, based on some rules/definitions and the schedules of previous years, such as:

- Each degree is composed of several courses, which are taught according to the amount of hours/ type of class as defined by the course syllabus.
- Each course is defined as semestral or annual and inserted in a specific year of the degree.
- Each group of students attends the same degree, courses and classes, scheduled at the same time slots.
- The assignment of groups of students and faculty members to courses is already made (e.g. the group of students 1DA of LEM attends MATE1 taught by MPS and JPM).
- Weekly plan for each course and type of class (e.g., MATE1 is divided into 2 hours of theoretical class and other 2 hours of theoretical-practical class).
- Theoretical (T) classes can be shared by more than one faculty member.
- Most of the department classrooms don't possess limitations regarding their availability.

Course-based timetables are created twice a year – before the beginning of each semester, therefore it is not a process that has to be done with short computational times. The main problems that come up with creating timetables manually are:

- The process can last several days.
- The quality of the timetables can be compromised due to deadlines and the experience of the timetabling developers.
- The result is influenced by the initial method used.
- Each school has different constraints, so there is no general algorithm.
- Dealing with conflicts, which can require previous assignments to be changed consecutively.

In DEM-ISEP there are some pre-assignments defined, regarding the service distribution of each faculty member. First, the number of students enrolled in each course are considered to divide

them into several groups of students. This division considers the maximum number each type of class allows. For example, theoretical classes allow a maximum of 100 students; if a certain course has 200 students enrolled, the number of theoretical classes must be two. The following table shows how the distribution is made for each course, according to the number of students enrolled, i.e., the number of classes needed per course.

Table 7 - Service distribution per course

	Number of classes		
	T	**TP**	**PL**
ALGAN	1	5	0
APROG	1	0	10
CMATE-M	1	0	11
DEGER	0	6	11
FSIAP	1	8	0
IENG1	1	0	9

Then, each faculty member is assigned to the respective courses, number of classes and type of class. The following table shows the service distribution for ALGAN, where it is possible to understand that, for example, SMA teaches one TP class while MGM teaches the theoretical class and two TP classes. The total of hours for classes T is 1, and for classes TP is 5, corresponding to the information disposed on table 7. The complete table is presented in ATTACHMENT A.

Table 8 - Service distribution for ALGAN

Professor	**Number of classes**		
	T	TP	PL
MGM	1	2	0
ASB	0	2	0
SMA	0	1	0

Also, considering the characteristics of each room and the specific needs of each class, to each classroom is attributed a type, according to its capacity and features. Table 9 is an extract of the real data information to illustrate.

Table 9 - Designation of each class and its capacity

Classroom	**Capacity**	**Type**
F202	32	TP
F208	50	TP
F209	40	TP
F214	24	PL
F216	24	PL
F218	24	PL
F341	98	T
F342	104	T
I201	110	T

3.2. Mathematical Model Adapted to the Case-Study

As mentioned above, an adaptation was conducted considering the characteristics and needs of the mechanical engineering department at ISEP. The model adapted is an integer programming formulation that considers seven main parameters, two decision variables, three sets of constraints, and an objective function to minimize the penalties related to the period of the day.

3.2.1. General features of the model

Considering the characteristics of the environment of the case study, and the actual method used to construct timetables at ISEP-DEM, the general parameters of the model chosen to adapt were kept (I, J, K, L, M, N) and added one more (T), considering the type of class.

- I set of days of the week (from Monday to Friday).
- J set of timetable periods of the day (from 8 a.m. to 8 p.m.).
- K set of DEM degrees and respective year (for example, if K is equal to 11, then the first number represents the LEM degree and the second number represents the first school year).
- L set of faculty members.
- M set of courses.
- N set of classrooms available.
- T set of type of classes (T, TP, PL, or OT).

The parameter T was added considering that in the future, the requirements might change, therefore, creating a flexible and adaptable algorithm is recommended. In fact, according to the preferences of the department, class lectures can be schedule by type, i.e., it facilitates grouping these only in the morning/afternoon. It is important to mention that the term 'class' does not represent the usual group of students that shares a whole schedule per week. On this case, and following the idea of the original article, that term 'class' is represented as 'group of students' which is attending one same course, at the same time, for example. When the term 'class' is mentioned, it is referring to the lecture class.

The special features that englobe new sets of parameters, based on the main ones, were adapted considering the data available to test the model and to facilitate the process of fitting the constraints. Therefore, several combinations between parameters were further performed to adapt the model: the parameters are associated in a way that each professor can only be assigned to the respective course and consequently to the classrooms associated with that type of class. For example, if Professor 'MGM' is teaching 'ALGAN', combinations with other courses are not possible. To do so, blocks of students (as groups of students) were created, i.e., for block X, there is only one professor, and one course. Each association between the group of students, faculty members, courses, classrooms, and type of lecture class represents one block. Each block represents temporally one hour in the timetable. To that hour, the parameters will be defined according to the possible combinations. Then, what can be variable regarding the rest of the parameters is the day and period of the day of lecture, and also the several classrooms where it can be assigned. These classrooms are already filtered, according to the type of lecture class for block 0 – T, TP, or PL.

Considering that the information is aggregated according to *k*, which is the key for each class (block), the parameters were defined following this logic, i.e., while accessing information related to some

value of the other parameters (classrooms, professors, course, type of class), the value of *k* is unique for each one of the combinations. Therefore, the following parameters emerge:

- I_k – day *i* available for degree *k* (there are some years of some degrees that have classes on Saturdays, for example).
- J_k – time period *j* available for degree *k* (related to the academic year of each degree – alternating courses in the morning and afternoon).
- J_{ki} – time period *j* of the day *i* available for degree *k.*
- L_k – faculty member of degree *k.*
- L_{km} – faculty member of degree *k,* teaching course *m*.
- L_{kmt} – faculty member of degree *k,* teaching a class of type *t* for course *m*.
- K_l – degrees for which professor *l* offers some course (for example, professor 'ABC' can teach more than one degree).
- M_k – courses of degree *k* (for example, the courses are the ones mentioned above for *k* = 11: ALGAN, APROG, CMATE-M, DEGER, FSIAP, IENG1).
- M_{kl} – courses of degree *k,* taught by professor *l.*
- M_{kn} – courses of degree *k,* taught in classroom *n.*
- T_{km} – type of classes for course *m* of degree *k* (this information is obtained through the service distribution mentioned in table 7).
- T_{kml} – type of classes for course *m* taught by professor *l* of degree *k* (this information is obtained through the service distribution mentioned in table 8).
- T_{kmn} – type of classes for course *m* of degree *k*, taught in classroom *n.*N_k – classroom *n* that fits for degree *k.*
- N_{kt} – classroom *n* that fits for the type of class *t* of degree *k* (this information is obtained through classroom definition mentioned in table 9).
- N_{kmt} – classroom *n* that fits for the type of class *t* for course *m* of degree *k.*
- c_{kmlt} – number of classes (repetitions) of type *t* for course *m* of degree *k*, taught by professor *l.*
- p_{ki} – periods available for each year of degree *k* on day *i* (for example, for the first year of the bachelor's degree in mechanical engineering – *k* = 11, the periods available are 5, from 8 a.m. until 1 p.m.).
- h_{kmt} – duration (hours) for the type of class *t* for course *m* of degree *k.*

One main set of binary variables is used. This variable has included the new general parameter added and is represented by $x_{i,j,k,m,l,t,n,c}$. An auxiliary binary variable is used, with the same indexes, $y_{i,j,k,m,l,t,n,c}$ and is used on the completeness and consecutiveness constraints.

To illustrate how the adapted model works, follows an additional example. Supposing professor 'ABC' is teaching two lab ('PL') classes of course 'VWXYZ', if the variable, x[0, 0, 11, 'VWXYZ', 'ABC', 'PL', 'F101',1] is equal to 1, this means that on Monday at 8 am, teacher 'ABC' is teaching the a block of one hour of the first of the two ('PL') classes of 'VWXYZ' in the room 'F101'. This variable represents an hour block.

3.2.2. Constraints for the IP model

The constraints for the adapted model include uniqueness constraints – to guarantee the non-existence of conflicts; completeness constraints – to ensure the timetable is completed according to the required characteristics; and consecutiveness constraints – so that multiperiod classes are represented by consecutive blocks.

- The first constraint – equation (15), ensures the uniqueness for each faculty member, i.e., every professor is assigned at most to one course, one class and classroom, on a given period of the day of a given day:

$$\sum_{m \in M_{kl}} \sum_{t \in T_{kml}} \sum_{n \in N_{kt}} \sum_{c=1}^{c_{kmlt}} x_{i,j,k,m,l,t,n,c} \leq 1, \forall k \in K, \forall i \in I_k, \forall j \in J_{ki}, \forall l \in L_k \quad (15)$$

- Equation (16) guarantees that every classroom may be assigned at most one course, one faculty member and one group of students on a given period of day of a given day:

$$\sum_{m \in M_{kn}} \sum_{t \in T_{kmn}} \sum_{l \in L_{kmt}} \sum_{c=1}^{c_{kmlt}} x_{i,j,k,m,l,t,n,c} \leq 1, \forall k \in K, \forall i \in I_k, \forall j \in J_{ki}, \forall n \in N_k \quad (16)$$

- Considering the structure of the parameters created, the original set of completeness constraints was adapted, and the equations were merged due to the redundancy, obtaining equations (17) and (18), that ensure simultaneously the development of the timetables considering both the right amount of teaching periods for each group of students and the required teaching periods for each course. The first equation of this set guarantees that each class of one hour is scheduled only one time. On equation (17), the auxiliary variable combined with the consecutiveness constraint – equation (18), ensures that each class that needs more than one teaching period, is present in the timetable in the right amount (h_{kmt}).

$$\sum_{i \in I_k} \sum_{n \in N_{kt}} \sum_{j=1}^{j=p_{ki}-h_{kmt}+1} y_{i,j,k,m,l,t,n,c} = 1, \forall k \in K, \forall m \in M_k, \forall l \in L_{km}, \forall t \in T_{kml}, \forall c \leq c_{kmlt} \quad (17)$$

- To ensure that multiperiod classes (classes with more than one hour) are consecutive, i.e., in periods of the day with no break between them, equation (18) was adapted:

$$\sum_{j_v=j}^{j+h_{kmt}} x_{i,jv,k,m,l,t,n,c} \geq h_{kmt} * y_{i,j,k,m,l,t,n,c} \quad \forall k \in K, \forall i \in I_k, \forall m \in M_k, \forall l \in L_{km}, \forall t \in T_{kml}, \forall n \in N_{kt}, \forall j \in J_{ki} \wedge j \leq p_{ki} - h_{kmt}, \forall c \leq c_{kmlt} \quad (18)$$

It was also necessary to add a constraint of non-overlapping classes, i.e., the same class taking different classes simultaneously. For example, for ALGAN, the service distribution shows that there

is one T class for the whole five TP. None of those TP classes can be scheduled at the same time as the T class.

- C_k – classes of degree *k* which aggregate groups of students of other type of classes of the same course. (this information is obtained through the service distribution mentioned in table 8).

$$C_{11} = \{(\text{ALGAN}, \text{MGM}, \text{T}, 1), (\text{APROG}, \text{JSM}, \text{T}, 1), \dots, (\text{IENG1}, \text{LMD}, \text{T}, 1)\}$$

- R_{kmlt} – classes with a hierarchical relation with class of course *m* of type *t* taught by professor *l* of degree *k* (this information is obtained through the service distribution mentioned in table 8).

$$R_{11,ALGAN,MGM,T} = \{(\text{ALGAN}, \text{MGM}, \text{TP}, 1), (\text{ALGAN}, \text{MGM}, \text{TP}, 2), \dots, (\text{ALGAN}, \text{SMA}, \text{TP}, 2)\}$$

$$\sum_{n \in N_{kmt}} \sum_{j=1}^{j=p_{ki}-h_{kmt}+1} y_{i,j,k,m,l,t,n,c} + \sum_{n \in N_{kmtr}} \sum_{j=1}^{j=p_{ki}} y_{i,j,k,m,lr,tr,n,cr} \leq 1, \forall k \in K, \forall i \in I_k, \forall (m,l,t,c) \in C_k, \forall (m,lr,tr,cr) \in R_{kmlt}$$

3.2.3. Objective function for the IP model

Considering the objective function, it intends to minimize the penalties associated with the period when the class is taught, as equation (20) shows.

$$minimize \sum_{k \in K} \sum_{i \in I_k} \sum_{j \in J_{ki}} \sum_{m \in M_k} \sum_{t \in T_{mk}} \sum_{l \in L_{kmt}} \sum_{n \in N_{kt}} x_{i,j,k,m,l,t,n} * j \quad (20)$$

This parameter *j* is working as a method to compact the timetable, in a scale from zero to four, i.e., it is rated as most preferred periods of the day with the value zero and the less preferred periods with the maximum value. The function will work as a minimization of penalties. The table below illustrates how this parameter is working.

Table 10 - Illustrative example of the cost parameter

Period of the day	Penalty value
08h10 - 09h00	0
09h10 - 10h00	1
10h10 – 11h00	2
11h10 – 12h00	3
12h10 – 13h00	4

3.3. Data used on the computational tests

To implement and validate the adapted model, real data were used performing the tests. This data was provided by ISEP, stored on Excel files (*df_salas.xlsx* and *ds_curso.xlsx*), by the mechanical engineering department. The data was combined in order to approach the problem according to the characteristics of the case-study.

This way, the information provided shows that on the first semester of the first academic year, the bachelor's degree in mechanical engineering (LEM) consists of 6 courses (ALGAN, APROG, CMATE-M, DEGER, FSIAP, IENG1). Twenty-five faculty members are assigned by the department. Concerning the types of classrooms, among the 20 available, 12 of them are regular classrooms (to theoretical and theoretical-practical classes) and the remaining 8 are intended for laboratory classes. Below, table 11 summarizes the information regarding the courses and the respective classes according to the type.

Table 11 - Curriculum for the first semester of the first-year students of mechanical engineering

Courses	Code	Periods		
		T	TP	PL
Linear Algebra and Analytic Geometry	ALGAN	2	2	
Algorithmic and Programming	APROG	2		2
Materials Science	CMATE-M	2		2
Advanced Engineering Drawing	DEGER		2	2
Applied Physics	FSIAP	1	2	
Introduction to Engineering I	IENG1	1		2

3.3.1. Data organization

An intensive analysis of the model was conducted to understand the best way to organize the available data so that the adapted model could be validated. Therefore, the parameters are grouped in a set of triplets (course, faculty member, and group of students), a set of pairs (day, period of the day), and a single set relative to the classrooms. In addition, one more parameter was added as index of the binary variable x: t; t is related to the type of class, as mentioned above. This index was added considering that in the future, the requirements might change, therefore, creating a flexible and adaptable algorithm will be very advantageous. In fact, according to the preferences of the department, class lectures can be schedule by type, i.e., it facilitates grouping these only in the morning/afternoon.

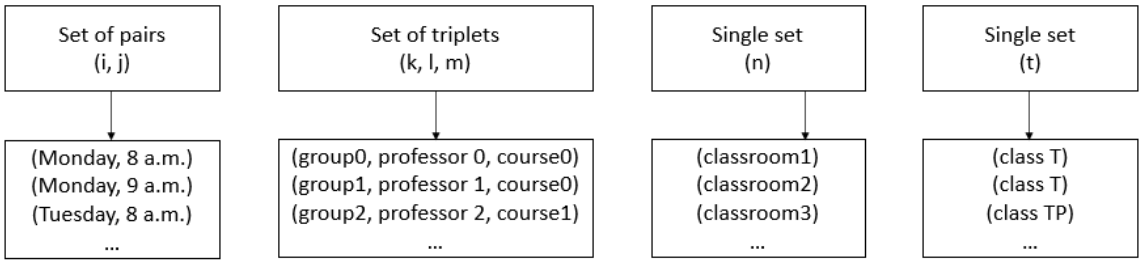


Figure 4 - Grouping parameters into sets

3.4. Structure of the computational application

On this chapter it is described how the data was grouped as well as the programming language used and the intersection between both.

3.4.1. Python + Pyomo + Excel

Python is a programming language that can be used for a wide variety of such as optimization, data analysis, machine learning, and others, with a vast number of libraries and resources available. Additionally, Python offers easy integration with a wide range of third-party libraries. This is particularly useful when you need to combine optimization with other tasks, such as data analysis or visualization. Particularly in this case, Python, with the Pyomo library, allows you to model and solve optimization problems flexibly, customizing the model and algorithm to match the specific needs of the problem, such as combining parameters to make data easier to read. To solve the optimization problems, CPLEX V22.1.0 was used.

In the process of adapting the model to the DEM problem, there was a challenge of accessing specific information about groups of students, courses, faculty members, classrooms, class types, and hours of the class. In order to simplify this process and make it more efficient in terms of coding, a special parameter that combines several key information into a single access point was introduced – *bloco_sala.*

The data was defined, as observable below on table 10, mainly using lists. Those lists contain tuples inside, where the data is stored and remains unaffected. For example, when we combine the group of student's indexes and classrooms ID, the result is a list with tuples that shows which classes are allocated to which classrooms, as the figure 5 below shows. The code used to define the constraints and objective function for the model can be found on ATTACHMENT B.

```
turmas_salas = [(i,salas[j][0]) for i in range(0,len_turmas) for j in range(0,len_salas) for k in range(0,len_turmas_UC) \
                if (salas[j][1] == turmas_UC[k][4] and i == turmas_UC[k][0])]
print(turmas_salas[0:20])
len_turmas_salas = len(turmas_salas)
print(len_turmas_salas)
```

```
[(0, 'F341'), (0, 'F342'), (0, 'I201'), (0, 'I301'), (1, 'F202'), (1, 'F203'), (1, 'F204'), (1, 'F207'), (1, 'F208'), (1,
'F209'), (1, 'F317'), (1, 'F319'), (2, 'F202'), (2, 'F203'), (2, 'F204'), (2, 'F207'), (2, 'F208'), (2, 'F209'), (2, 'F317
'), (2, 'F319')]
500
```

Figure 5 - Code extract that combines parameters into a list.

Table 12 - Special parameters and the respective description

Parameter (list)	Description
turma_sala_tipo	List that associates groups of students with available classrooms, according to the type of class.
hora_aula	List that contains information about how long lasts each class for each group of students (H_k).
turma_UC	List that combines information about courses, and student groups.
turma_prof	List that combines information of professors and student groups.
bloco_sala	List that includes information about groups of students, professors, courses, classrooms, type of class.

4. RESULTS AND DISCUSSION

In this chapter, the results obtained from this research are presented. During the development of this study, the timetabling problem in a university environment is explored and some advanced methodologies to resolve crucial issues that affect the optimization and efficiency of this process were applied, such as mixed-integer programming, using subsets of variables (reducing the number of variables to keep them manageable). The focus was on the search for practical and effective solutions, with the aim of improving the allocation of resources and the distribution of classes, minimizing the time this task usually takes.

It is also presented a comprehensive analysis of the results obtained and discussion of them, from the adapted optimization model, describing how the penalties associated with different combinations of classes and resources affect the resulting allocations and schedules. This critical analysis allows the evaluation of the model's performance in terms of efficiency, quality and practical feasibility.

4.1. Results obtained

For the first semester of the first year, the results were used to create a map showing how many classes per type are required for each course, who is teaching them, and at what time and day, in which classroom. An example follows – table 13, which shows a minor extract of the results organized according to each course. The columns numbered from 1 to 4 represent possible groups of students (as classes), i.e., group 1 would be assigned TP1 and PL1 of DEGER, and group 2 would attend to TP1 as well, but PL2 instead. The complete results for each course are presented on APPENDIX A.

Table 13 - Partial example of the scheduling map per course

<table>
<tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th></tr>
<tr><td rowspan="2">DEGER:
2h TP
2h PL</td><td colspan="2">TP 1; Friday 8 a.m. - 10 a.m.
Professor: AGS; classroom F207</td><td colspan="2">TP 2; Friday 8 a.m. - 10 a.m.
Professor: LMT; classroom F208</td></tr>
<tr><td>PL 1; Tuesday 8 a.m. - 10 a.m.
Professor: AGS; classroom F221</td><td>PL 2; Thursday 8 a.m. - 10 a.m.
Professor: AGS; classroom F226</td><td>PL 3; Monday 8 a.m. - 10 a.m.
Professor: LMT; classroom F214</td><td>PL 4; Tuesday 8 a.m. - 10 a.m.
Professor: LMT; classroom F214</td></tr>
</table>

In total, 128 blocks of classes were assigned. The distribution was homogenous, but as the objective function intended, the classes were scheduled mainly for the first two periods of the day. On Tuesdays and Fridays, more classes are assigned, with 29 and 30 blocks, respectively. This distribution highlights that new constraints should be introduced in order to balance better the use of the periods of time. Table 14 illustrates this distribution.

Table 14 - Distribution of the blocks

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
8H - 9H	13	14	11	11	15
9H - 10H	12	13	11	11	15
10H - 11H		1			
11H - 12H		1			
12H-13H					
TOTAL/DAY	25	29	22	22	30
TOTAL/WEEK	128				

Furthermore, scheduling maps for each day of the week were created. The following table shows an example of these maps. the complete results can be consulted on APPENDIX B.

Table 15 - Scheduled map for Wednesday (extract)

	Wed			
8-9	APROG, JSM, PL, F218	CMATE-M, NAL, PL, F225	DEGER, OCF, PL, F226	DEGER, EFM, PL, F221
9-10	APROG, JSM, PL, F218	CMATE-M, NAL, PL, F225	DEGER, OCF, PL, F226	DEGER, EFM, PL, F221
10-11				
11-12				
12-13				

4.2. Discussion of the results

Considering the results, there are several important points to outline:

- All blocks are present in the schedule as many times as necessary for the duration of the classes.
- There are no overlaps: there are no rooms to be occupied at the same time; the same teacher does not teach more than one class in each period.
- The distribution of classes is very homogeneous in the first periods of the day, given that in the objective function the penalty is related to the periods of the day (the later, the greater the penalty).
- Consecutiveness of multiperiod classes is guaranteed.
- Each class is scheduled according to the type of class using a respective classroom to that same type.

It is also important to highlight that for the same course, different types of classes are not scheduled at the same time (when associated), allowing students to attend all the necessary classes. For example, the following table 16 shows the scheduling of ALGAN. According to the data provided, there are, in total, 5 different groups of students attending this course: all of them attend the same theoretical class and each group has one different TP class assigned. The first line of the table indicates this specific need for each student attending each course. In this case, it is possible to notice that none of the TP classes is scheduled on the same day as the T class; although T2, T4, and TP5 are scheduled on the same day, that does not generate any conflict.

Table 16 - Scheduled classes of ALGAN

ALGAN: 2h T; 2h TP				
T 1; Friday 8 a.m. - 10 a.m. Professor: MGM; classroom: F342				
TP 1; Monday 8 a.m. - 10 a.m. Professor: MGM; classroom F202	TP 2; Tuesday 8 a.m. - 10 a.m. Professor: MGM; classroom F202	TP 3; Monday 8 a.m. - 10 a.m. Professor: ASB; classroom F204	TP 4; Tuesday 8 a.m. - 10 a.m. Professor: ASB; classroom F209	TP 5; Tuesday 8 a.m. - 10 a.m. Professor: SMA; classroom F203

The objective function compacts the schedule, as it can be seen above, and the other periods available could be used to solve the overlapping problem that students might find choosing the blocks to complete their schedule.

Besides this, there were no pre-assignment constraints tested, but a set of these required constraints can be easily added, according to pre-defined allocations. For example, if professor 'ABC' must teach a course in 'VWXYZ' on Monday, at 8 a.m., the variable x [0, 0, 0, 'VWXYZ', 'ABC', 'T', 'I201'] is equal to 1, varying the parameters according to the group of students, type of class and classroom related to the course and faculty member.

The adapted model to this mixed integer problem carried 25126 constraints and 50001 binary variables, while the number of non-zeros was 171861. The computational time was about 50 seconds, but this varied from problem to problem: the more restrictions added, the more it took to get feasible results.

5. CONCLUSION

This chapter summarizes the work carried out, the objective of the dissertation project accomplished, as well as the difficulties identified. Afterwards, the limitations are explained, as well as a brief description of future research.

5.1. Final conclusions

The proposed project aimed to develop an algorithm that supported the construction of timetables in ISEP, since it is a complex and tedious task performed manually, most of the time based on schedules provided in the previous years.

To solve this problem, an extensive review of the literature related to the construction of university timetables was conducted, which made it possible to identify key challenges such as classroom constraints, teacher and student preferences, curriculum requirements, and the inherent complexity of this task.
Consequently, an algorithm was built adapting an existing one, forcing the development of a data structure that considers some of the specific needs of DEM-ISEP. Taking into account the work developed, it is now easier to identify which type of information is relevant to adapt to the data structure. This algorithm was implemented in a controlled environment and tested to validate it and compare the results with traditional manual solutions.
As a result, a map of all the 128 blocks of one-hour classes were distributed among the week, scheduled from 8 a.m. till 1 p.m., presenting the faculty member, the classroom assigned and the type of class according to each course.
This research presents feasible solutions that need improvement and through that correspond to the demanding conditions and restrictions imposed by ISEP. It is important to highlight that the computational results were obtained in less than one minute, which is a great decrease in the time, compared to the manual work previously done.

5.2. Limitations and future research

Solving the UCTP can be very challenging. It is necessary to analyse in detail the conditions evolving the environment where it is wanted to be implemented. On this case study, the biggest focus was the data structure used in the computational application (using Python), and solving the problem outlined the limitations associated with this problem.

The main problem is related to the complexity of these models themselves. The problem becomes more complex as the number of restrictions, courses, teachers and rooms increases, and developing more efficient algorithms to solve these complex problems is needed. Adding restrictions and constraints is a big challenge and it was not always satisfied.

In practice, available data may be incomplete or uncertain, which may affect the quality of solutions. In this case, the data was incomplete, which did not allow, for example, to assure the pre-assignment constraints.

Faculty members and also students may have specific preferences regarding schedules. Surveys could be conducted as future research to model these preferences accurately. These surveys could include the evaluation of the satisfaction of the actual timetables used.

Besides that, the distribution of the workload should be studied, and balanced equally among both faculty members and students. Future research could explore methods to optimize the workload and the duration of each class, for example.

Considering the conscious usage of resources, since each classroom has different characteristics, it would be interesting to optimize the physical spaces, in order to save energy, taking into account the number of students. As an example, classes with capacity for 100 students should not be assigned to classes with 20 students.

To sum up, the timetabling problem in a university context is a complex and challenging problem with numerous potential areas of research. Developing more sophisticated models and considering the needs and preferences of both faculty members and students are fundamental aspects of future research in this area.

LITERATURE REFERENCES

[1] Akif Bakır, M., & Aksop, C. (2008). A 0-1 INTEGER PROGRAMMING APPROACH TO A UNIVERSITY TIMETABLING PROBLEM. In *Hacettepe Journal of Mathematics and Statistics* (Vol. 37, Issue 1).

[2] Aziz, N. L. A., & Aizam, N. A. H. (2018). A brief review on the features of university course timetabling problem. *AIP Conference Proceedings*, *2016*. https://doi.org/10.1063/1.5055403

[3] Bashab, A., Ibrahim, A. O., AbedElgabar, E. E., Ismail, M. A., Elsafi, A., Ahmed, A., & Abraham, A. (2020). A systematic mapping study on solving university timetabling problems using meta-heuristic algorithms. In *Neural Computing and Applications* (Vol. 32, Issue 23, pp. 17397–17432). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/s00521-020-05110-3

[4] Birbas, T., Daskalaki, S., & Housos, E. (2009). School timetabling for quality student and teacher schedules. *Journal of Scheduling*, *12*(2), 177–197. https://doi.org/10.1007/s10951-008-0088-2

[5] Burke, E. K., & Petrovic, S. (2002). *Recent research directions in automated timetabling*. www.elsevier.com/locate/dsw

[6] Carter, M. W., & Laporte, G. (1998). *Recent Developments in Practical Course Timetabling*.

[7] Carter, M. W., & Laporte, O. (1996). *Recent Developments in Practical Examination Timetabling*.

[8] Ceschia, S., di Gaspero, L., & Schaerf, A. (2012). Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers and Operations Research*, *39*(7), 1615–1624. https://doi.org/10.1016/j.cor.2011.09.014

[9] Chen, M. C., Sze, S. N., Goh, S. L., Sabar, N. R., & Kendall, G. (2021). A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. *IEEE Access*, *9*, 106515–106529. https://doi.org/10.1109/ACCESS.2021.3100613

[10] Daskalaki, S., & Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, *160*(1), 106–120. https://doi.org/10.1016/j.ejor.2003.06.023

[11] 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. https://doi.org/10.1016/S0377-2217(03)00103-6

[12] Dimopoulou, M., & Miliotis, P. (2001). *Implementation of a university course and examination timetabling system*. www.elsevier.com/locate/dsw

[13] Firdaus Khair, A., Makhtar, M., Mazlan, M., Mohamed, M. A., Nordin, M., & Rahman, A. (2018). A study on university course and exam timetabling problems and methods: an optimization survey. In *International Journal of Engineering & Technology* (Vol. 7, Issue 2).

[14] Gülcü, A., & Akkan, C. (2020). Robust university course timetabling problem subject to single and multiple disruptions. *European Journal of Operational Research*, *283*(2), 630–646. https://doi.org/10.1016/j.ejor.2019.11.024

[15] Johnes, J. (2015). Operational research in education. In *European Journal of Operational Research* (Vol. 243, Issue 3, pp. 683–696). Elsevier B.V. https://doi.org/10.1016/j.ejor.2014.10.043

[16] Kingston, J. H. (2014). Timetable construction: The algorithms and complexity perspective. *Annals of Operations Research*, *218*(1), 249–259. https://doi.org/10.1007/s10479-012-1160-z

[17] McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., di Gaspero, L., Qu, R., & Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, *22*(1), 120–130. https://doi.org/10.1287/ijoc.1090.0320

[18] Mehrad, A., & Tahriri Zangehen, M. (2019). Comparison between Qualitative and Quantitative Research Approaches: Social Sciences. *International Journal For Research In Educational Studies*, *5*(7).

[19] Méndez-Díaz, I., Zabala, P., & Miranda-Bront, J. J. (2016). An ILP based heuristic for a generalization of the post-enrollment course timetabling problem. *Computers and Operations Research*, *76*, 195–207. https://doi.org/10.1016/j.cor.2016.06.018

[20] Mikkelsen, R., & Holm, D. S. (2022). A parallelized matheuristic for the International Timetabling Competition 2019. *Journal of Scheduling*. https://doi.org/10.1007/s10951-022-00728-8

[21] Mirhassani, S. A., & Habibi, F. (2013). Solution approaches to the course timetabling problem. *Artificial Intelligence Review*, *39*(2), 133–149. https://doi.org/10.1007/s10462-011-9262-6

[22] Mokhtari, M., Vaziri Sarashk, M., Asadpour, M., Saeidi, N., & Boyer, O. (2021). Developing a Model for the University Course Timetabling Problem: A Case Study. *Complexity*, *2021*. https://doi.org/10.1155/2021/9940866

[23] Palma, C. D., & Bornhardt, P. (2020). Considering section balance in an integer optimization model for the curriculum-based course timetabling problem. *Mathematics*, *8*(10), 1–12. https://doi.org/10.3390/math8101763

[24] Pandey, J., & Sharma, A. K. (2016). *Survey on University Timetabling Problem*.

[25] Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. *Annals of Operations Research*, *239*(1), 3–38. https://doi.org/10.1007/s10479-014-1688-1

[26] Prabodanie, R. A. R. (2017). An Integer Programming Model for a Complex University Timetabling Problem: A Case Study. *Industrial Engineering and Management Systems*, *16*(1), 141–153. https://doi.org/10.7232/iems.2017.16.1.141

[27] Rappos, E., Thiémard, E., Robert, S., & Hêche, J. F. (2022). A mixed-integer programming approach for solving university course timetabling problems. *Journal of Scheduling*, *25*(4), 391–404. https://doi.org/10.1007/s10951-021-00715-5

[28] Rezaeipanah, A., Matoori, S. S., & Ahmadi, G. (2021). A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. *Applied Intelligence*, *51*(1), 467–492. https://doi.org/10.1007/s10489-020-01833-x

[29] Rudová, H., & Murray, K. (2002). *University Course Timetabling with Soft Constraints*.

[30] Sarin, S. C., Wang, Y., & Varadarajan, A. (2010). A university-timetabling problem and its solution using Benders' partitioning - A case study. *Journal of Scheduling*, *13*(2), 131–141. https://doi.org/10.1007/s10951-009-0157-1

[31] Schimmelpfeng, K., & Helber, S. (2007). Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, *29*(4), 783–803. https://doi.org/10.1007/s00291-006-0074-z

[32] Siddiqui, A. W., & Arshad Raza, S. (2021). A general ontological timetabling-model driven metaheuristics approach based on elite solutions. *Expert Systems with Applications*, *170*. https://doi.org/10.1016/j.eswa.2020.114268

[33] Siddiqui, A. W., Raza, S. A., & Tariq, Z. M. (2018). A web-based group decision support system for academic term preparation. *Decision Support Systems*, *114*, 1–17. https://doi.org/10.1016/j.dss.2018.08.005

[34] Song, T., Liu, S., Tang, X., Peng, X., & Chen, M. (2018). An iterated local search algorithm for the University Course Timetabling Problem. *Applied Soft Computing Journal*, *68*, 597–608. https://doi.org/10.1016/j.asoc.2018.04.034

[35] Sørensen, M., & Dahms, F. H. W. (2014). A Two-Stage Decomposition of High School Timetabling applied to cases in Denmark. *Computers and Operations Research*, *43*(1), 36–49. https://doi.org/10.1016/j.cor.2013.08.025

[36] Sylejmani, K., Gashi, E., & Ymeri, A. (2022). Simulated annealing with penalization for university course timetabling. *Journal of Scheduling*. https://doi.org/10.1007/s10951-022-00747-5

[37] Tan, J. S., Goh, S. L., Kendall, G., & Sabar, N. R. (2021). A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, *165*. https://doi.org/10.1016/j.eswa.2020.113943

[38] Valouxis, C., & Housos, E. (2003). Constraint programming approach for school timetabling. In *Computers & Operations Research* (Vol. 30). www.sciencedirect.comwww.elsevier.com/locate/dsw

[39] Vermuyten, H., Lemmens, S., Marques, I., & Beliën, J. (2016). Developing compact course timetables with optimized student flows. *European Journal of Operational Research*, *251*(2), 651–661. https://doi.org/10.1016/j.ejor.2015.11.028

APPENDIX A – SCHEDULED MAP FOR EACH COURSE

DEGER: 2h TP; 2h PL										
TP 1; Wednesday 8 a.m. - 10 a.m. Professor: AGS; classroom F204		TP 2; Wednesday 8 a.m. - 10 a.m. Professor: LMT; classroom F317		TP 3; Tuesday 8 a.m. - 10 a.m. Professor: OCF; classroom F319		TP 4; Monday 8 a.m. - 10 a.m. Professor: EFM; classroom F203		TP 5; Friday 8 a.m. - 10 a.m. Professor: JOB; classroom F209		TP 6; Wednesday 8 a.m. - 10 a.m. Professor: JFS; classroom F208
PL 1; Tuesday 8 a.m. - 10 a.m. Professor: AGS; classroom F216	PL 2; Monday 8 a.m. - 10 a.m. Professor: AGS; classroom F225	PL 3; Tuesday 8 a.m. - 10 a.m. Professor: LMT; classroom F214	PL 4; Thursday 8 a.m. - 10 a.m. Professor: LMT; classroom F218	PL 5; Wednesday 8 a.m. - 10 a.m. Professor: OCF; classroom F221	PL 6; Thursday 8 a.m. - 10 a.m. Professor: OCF; classroom F214	PL7; Thursday 8 a.m. - 10 a.m. Professor: EFM; classroom F322	PL8; Friday 8 a.m. - 10 a.m. Professor: EFM; classroom F218	PL9; Wednesday 8 a.m. - 10 a.m. Professor: JOB; classroom F214	PL10; Tuesday 8 a.m. - 10 a.m. Professor: JOB; classroom F322	PL11; Thursday 8 a.m. - 10 a.m. Professor: JFS; classroom F224

APROG: 2h T; 2h PL									
T 1; Tuesday 8 a.m. - 10 a.m. Professor: JSM; classroom I301									
PL 1; Friday 8 a.m. - 10 a.m. Professor: JSM; classroom F221	PL 2; Wednesday 8 a.m. - 10 a.m. Professor: JSM; classroom F322	PL 3; Monday 8 a.m. - 10 a.m. Professor: JSM; classroom F216	PL 4; Wednesday 10 a.m. - 12 a.m. Professor: JSM; classroom F221	PL 5; Thursday 8 a.m. - 10 a.m. Professor: JSM; classroom F221	PL 6; Monday 8 a.m. - 10 a.m. Professor: RAL; classroom F221	PL 7; Monday 8 a.m. - 10 a.m. Professor: NVM; classroom F218	PL 8; Tuesday 8 a.m. - 10 a.m. Professor: NVM; classroom F226	PL 9; Friday 8 a.m. - 10 a.m. Professor: NVM; classroom F224	PL 10; Monday 8 a.m. - 10 a.m. Professor: GCN; classroom F224

CMATE-M: 2h T; 2h PL										
T 1; Friday 8 a.m. - 10 a.m. Professor: RBC; classroom I201										
PL 1; Friday 8 a.m. - 10 a.m. Professor: SMS; classroom F216	PL 2; Friday 8 a.m. - 10 a.m. Professor: PPS; classroom F225	PL 3; Thursday 8 a.m. - 10 a.m. Professor: PPS; classroom F226	PL 4; Wednesday 8 a.m. - 10 a.m. Professor: PPS; classroom F216	PL 5; Monday 8 a.m. - 10 a.m. Professor: PPS; classroom F214	PL 6; Thursday 8 a.m. - 10 a.m. Professor: FAF; classroom F216	PL 7; Tuesday 8 a.m. - 10 a.m. Professor: FAF; classroom F221	PL 8; Wednesday 8 a.m. - 10 a.m. Professor: FAF; classroom F218	PL 9; Friday 8 a.m. - 10 a.m. Professor: NAL; classroom F214	PL 10; Monday 8 a.m. - 10 a.m. Professor: NAL; classroom F226	PL 11; Wednesday 8 a.m. - 10 a.m. Professor: NAL; classroom F225

IENG1: 1h T; 2h PL								
T 1; Tuesday 8 a.m - 9 a.m. Docente: LMD								
PL 1; Monday 8 a.m. - 10 a.m. Professor: LMD; classroom F322	PL 2; Friday 8 a.m. - 10 a.m. Professor: LMD; classroom F226	PL 3; Tuesday 8 a.m. - 10 a.m. Professor: SCF; classroom F218	PL 4; Thursday 8 a.m. - 10 a.m. Professor: SCF; classroom F225	PL 5; Wednesday 8 a.m. - 10 a.m. Professor: ICT; classroom F224	PL 6; Tuesday 8 a.m. - 10 a.m. Professor: ICT; classroom F224	PL 7; Friday 8 a.m. - 10 a.m. Professor: ICT; classroom F322	PL 8; Tuesday 8 a.m. - 10 a.m. Professor: IAA; classroom F225	PL 9; Wednesday 8 a.m. - 10 a.m. Professor: IAA; classroom F226

FSIAP: 1h T; 2h TP							
T 1; Thursday 8 a.m. - 9 a.m. Professor: MPA; classroom I301							
TP 1; Friday 8 a.m. - 10 a.m. Professor: MPA; classroom F208	TP 2; Tuesday 8 a.m. - 10 a.m. Professor: MPA; classroom F317	TP 3; Wednesday 8 a.m. - 10 a.m. Professor: MPA; classroom F207	TP 4; Monday 8 a.m. - 10 a.m. Professor: JNP; classroom F208	TP 5; Wednesday 8 a.m. - 10 a.m. Professor: APA; classroom F203	TP 6; Friday 8 a.m. - 10 a.m. Professor: APA; classroom F207	TP 7; Thursday 8 a.m. - 10 a.m. Professor: APA; classroom F208	TP 8; Tuesday 8 a.m. - 10 a.m. Professor: APA; classroom F204

ALGAN: 2h T; 2h TP				
T 1; Wednesday 8 a.m. - 10 a.m. Professor: MGM; classroom: I201				
TP 1; Thursday 8 a.m. - 10 a.m. Professor: MGM; classroom F319	TP 2; Tuesday 8 a.m. - 10 a.m. Professor: MGM; classroom F208	TP 3; Tuesday 8 a.m. - 10 a.m. Professor: ASB; classroom F203	TP 4; Wednesday 8 a.m. - 10 a.m. Professor: ASB; classroom F319	TP 5; Wednesday 8 a.m. - 10 a.m. Professor: SMA; classroom F202

APPENDIX B – SCHEDULED MAP FOR EACH DAY

	Mon												
8-9	ALGAN, MGM , TP, F202	ALGAN, ASB , TP, F204	APROG, JSM, T, I201	CMATE-M, PPS, PL, F216	CMATE-M, FAF, PL, F224	CMATE-M, NAL, PL, F322	DEGER, LMT, PL, F214	DEGER, EFM, PL, F221	DEGER, JFS, TP, F207	IENG1, SCF, PL, F225	IENG1, ICT, PL, F218	IENG1, IAA, PL, F226	FSIAP, MPA, T, I301
9-10	ALGAN, MGM , TP, F202	ALGAN, ASB , TP, F204	APROG, JSM, T, I201	CMATE-M, PPS, PL, F216	CMATE-M, FAF, PL, F224	CMATE-M, NAL, PL, F322	DEGER, LMT, PL, F214	DEGER, EFM, PL, F221	DEGER, JFS, TP, F207	IENG1, SCF, PL, F225	IENG1, ICT, PL, F218	IENG1, IAA, PL, F226	
10-11													
11-12													
12-13													

	Tue													
8-9	ALGAN, MGM, TP, F202	ALGAN, ASB, TP, F209	ALGAN, SMA, TP, F203	APROG, JSM, PL, F225	APROG, NVM, PL, F216	CMATE-M, SMS, PL, F226	CMATE-M, PPS, PL, F218	CMATE-M, NAL, PL, F224	DEGER, AGS, PL, F221	DEGER, LMT, PL, F214	DEGER, OCF, TP, F317	DEGER, JOB, PL, F322	FSIAP, APA, TP, F208	IENG1, LMD, T, I301
9-10	ALGAN, MGM, TP, F202	ALGAN, ASB, TP, F209	ALGAN, SMA, TP, F203	APROG, JSM, PL, F225	APROG, NVM, PL, F216	CMATE-M, SMS, PL, F226	CMATE-M, PPS, PL, F218	CMATE-M, NAL, PL, F224	DEGER, AGS, PL, F221	DEGER, LMT, PL, F214	DEGER, OCF, TP, F317	DEGER, JOB, PL, F322	FSIAP, APA, TP, F208	
10-11	APROG, JSM, PL, F225													
11-12	APROG, JSM, PL, F225													
12-13														

	Wed										
8-9	APROG, JSM, PL, F218	CMATE-M, PPS, PL, F214	CMATE-M, FAF, PL, F224	CMATE-M, NAL, PL, F225	DEGER, OCF, PL, F226	DEGER, EFM, PL, F221	DEGER, JOB, TP, F203	FSIAP, MPA, TP, F317	FSIAP, APA, TP, F208	IENG1, ICT, PL, F216	IENG1, IAA, PL, F322
9-10	APROG, JSM, PL, F218	CMATE-M, PPS, PL, F214	CMATE-M, FAF, PL, F224	CMATE-M, NAL, PL, F225	DEGER, OCF, PL, F226	DEGER, EFM, PL, F221	DEGER, JOB, TP, F203	FSIAP, MPA, TP, F317	FSIAP, APA, TP, F208	IENG1, ICT, PL, F216	IENG1, IAA, PL, F322
10-11											
11-12											
12-13											

	Thu										
8-9	APROG, JSM, PL, F322	APROG, NVM, PL, F216	APROG, GCN, PL, F225	CMATE-M, RBC, T, F341	DEGER, AGS, PL, F226	DEGER, JOB, PL, F214	DEGER, JFS, PL, F224	FSIAP, MPA, TP, F319	FSIAP, APA, TP, F202	IENG1, LMD, PL, F218	IENG1, SCF, PL, F221
9-10	APROG, JSM, PL, F322	APROG, NVM, PL, F216	APROG, GCN, PL, F225	CMATE-M, RBC, T, F341	DEGER, AGS, PL, F226	DEGER, JOB, PL, F214	DEGER, JFS, PL, F224	FSIAP, MPA, TP, F319	FSIAP, APA, TP, F202	IENG1, LMD, PL, F218	IENG1, SCF, PL, F221
10-11											
11-12											
12-13											

	Fri													
8-9	ALGAN, MGM, T, F342	APROG, JSM, PL, F216	APROG, RAL, PL, F218	APROG, NVM, PL, F225	CMATE-M, PPS, PL, F226	CMATE-M, FAF, PL, F221	DEGER, AGS, TP, F207	DEGER, LMT, TP, F208	DEGER, OCF, PL, F322	DEGER, EFM, TP, F204	FSIAP, MPA, TP, F202	FSIAP, JNP, TP, F209	FSIAP, APA, TP, F319	IENG1, LMD, PL, F224	IENG1, ICT, PL, F214
9-10	ALGAN, MGM, T, F342	APROG, JSM, PL, F216	APROG, RAL, PL, F218	APROG, NVM, PL, F225	CMATE-M, PPS, PL, F226	CMATE-M, FAF, PL, F221	DEGER, AGS, TP, F207	DEGER, LMT, TP, F208	DEGER, OCF, PL, F322	DEGER, EFM, TP, F204	FSIAP, MPA, TP, F202	FSIAP, JNP, TP, F209	FSIAP, APA, TP, F319	IENG1, LMD, PL, F224	IENG1, ICT, PL, F214
10-11															
11-12															
12-13															

ATTACHMENT A – SERVICE DISTRIBUTION

	Professor	Number of classes		
		T	TP	PL
ALGAN	MGM	1	2	0
	ASB	0	2	0
	SMA	0	1	0
APROG	JSM	1	0	5
	RAL	0	0	1
	NVM	0	0	3
	GCN	0	0	1
CMATE-M	RBC	1	0	0
	SMS	0	0	1
	PPS	0	0	4
	FAF	0	0	3
	NAL	0	0	3
DEGER	AGS	0	1	2
	LMT	0	1	2
	OCF	0	1	2
	EFM	0	1	2
	JOB	0	1	2
	JFS	0	1	1
FSIAP	MPA	1	3	0
	JNP	0	1	0
	APA	0	4	0
IENG1	LMD	1	0	2
	SCF	0	0	2
	ICT	0	0	3
	IAA	0	0	2

ATTACHMENT B – CODE

```
from pyomo.environ import *
import pyomo.environ as pyo
model = pyo.ConcreteModel()
model.KLMN = pyo.Set(initialize=bloco_sala)
model.I = pyo.Set(initialize=range(len_dias))
model.J = pyo.Set(initialize=range(len_horas))
model.K = pyo.Set(initialize=range(len_aulas))
model.x_index = model.I * model.J * model.KLMN
model.x = pyo.Var(model.x_index, within=pyo.Binary)
model.y = pyo.Var(model.x_index, within=pyo.Binary)
model.L = pyo.Set(initialize=profs)

# Constraints
model.uniqueness_constraint = pyo.ConstraintList()
for i in model.I: # Dia
   for j in model.J: # Hora
      for l in profs: # Prof
         expr = sum(model.x[i,j,bloco_sala[k][0],bloco_sala[k][1],l,bloco_sala[k][3],bloco_sala[k][4]]
                  for k in range(0,len_bloco_sala) \
                  if l == bloco_sala[k][2]) <= 1
         model.uniqueness_constraint.add(expr)

for i in model.I: # Dia
   for j in model.J: # Hora
      for k in range(0,len_salas):
         print(i,j,k,salas[k][0])
         expr =
sum(model.x[i,j,bloco_sala[k2][0],bloco_sala[k2][1],bloco_sala[k2][2],bloco_sala[k2][3],bloco_sala
[k2][4]] \
                  for k2 in range(0,len_bloco_sala)\
                  if salas[k][0] == bloco_sala[k2][4]) <= 1
```

```
        model.uniqueness_constraint.add(expr)


model.consecutiveness_constraint = pyo.ConstraintList()  # Consecutiveness Constraints
for i in model.I: # Dia
   for k in range(0,len_bloco_sala): # Aula
      if horas_aula[bloco_sala[k][0]] > 1:
         for j in model.J: # Hora
            if j <= (len_horas - horas_aula[bloco_sala[k][0]]):
               jhv = j + int(horas_aula[bloco_sala[k][0]])
               expr =
sum(model.x[i,t,bloco_sala[k][0],bloco_sala[k][1],bloco_sala[k][2],bloco_sala[k][3],bloco_sala[k][4
]] for t in range(j,jhv)) \
                        >= int(horas_aula[bloco_sala[k][0]]) *
model.y[i,j,bloco_sala[k][0],bloco_sala[k][1],bloco_sala[k][2],bloco_sala[k][3],bloco_sala[k][4]]
               model.consecutiveness_constraint.add(expr)


model.completeness_constraint = pyo.ConstraintList()
for k in range(0,len_horas_aula): # Aula
   if horas_aula[k] > 0:
      jhv = int(len_horas - horas_aula[k])
      expr =
sum(model.y[i,j,bloco_sala[k2][0],bloco_sala[k2][1],bloco_sala[k2][2],bloco_sala[k2][3],bloco_sala
[k2][4]] \
            for i in model.I for j in range(0,jhv) for k2 in range(0,len_bloco_sala)\
               if bloco_sala[k2][0] == k) == 1
      model.completeness_constraint.add(expr)


k_init = 0
k_next = 1
naula = 0
nbloco = 0
while nbloco < len_bloco_sala:
   naula = k_init
   nbloco = naula
```

```
    ik0_pointer = [naula]
    same_aula = True
    while same_aula:
        same_aula = False
        nbloco += 1
        if nbloco < len_bloco_sala:
            if bloco_sala[naula][0] == bloco_sala[nbloco][0]:
                same_aula = True
                ik0_pointer.extend([nbloco]) # Add block number pointer to list
    while (nbloco < len_hierq_aulas and hierq_aulas[nbloco] == hierq_aulas[k_init]):
        naula = nbloco
        ik1_pointer = [nbloco]
        nbloco += 1
        same_aula = True
        while same_aula:
            same_aula = False
            if nbloco < len_bloco_sala:
                if bloco_sala[naula][0] == bloco_sala[nbloco][0]:
                    same_aula = True
                    ik1_pointer.extend([nbloco]) # Add block number pointer to list
                    nbloco += 1
        for i in model.I:
            expr =
sum(model.y[i,j,bloco_sala[k][0],bloco_sala[k][1],bloco_sala[k][2],bloco_sala[k][3],bloco_sala[k][4
]] \
                    for k in ik0_pointer for j in model.J) + \

sum(model.y[i,j,bloco_sala[k][0],bloco_sala[k][1],bloco_sala[k][2],bloco_sala[k][3],bloco_sala[k][4
]] \
                    for k in ik1_pointer for j in model.J) <= 1
            model.uniqueness_constraint.add(expr)
    k_init = nbloco
    nbloco += 1
```

```
model.objective = pyo.Objective (expr =
sum(model.x[i,j,bloco_sala[k][0],bloco_sala[k][1],bloco_sala[k][2],bloco_sala[k][3],bloco_sala[k][4
]]*j \
                                for i in model.I for j in model.J for i in model.I for k in
range(0,len_bloco_sala)))
pyo.SolverFactory('cplex').solve(model).write()
```