Discrete Optimization and Decision Making - Final Project

Examination Timetabling Problem

Coppola Davide - VR479624
Di Grazia Domenico - VR481299
Sanfelici Alessia - VR487892

ACADEMIC YEAR 2022/2023

1 Introduction

This report aims at the illustration of the developed solution of the Examination Timetabling Problem. The project work consists in the formulation of this problem as an Integer Linear Programming model and the presentation and explanation of the used variables, constraints and objective function. The model was implemented in Python, using the Gurobi Optimizer package.
After a first presentation of the Examination Timetabling Problem, with an illustration of all the requirements that needed to be implemented, a precise description of the model is presented, with a further explanation of the choices made. Then, the following sections aim at presenting two variants of the basic problem: in the first one, we substituted the original objective function of the problem with a new equity measure, while in the second one, we added some additional constraints, to limit the possibilities in the creation of the timetable. Finally, the model's results are shown, with respect to a set of provided instances. The results are also compared to already available benchmarks.
The entire problem, together with the Python code, the test instances and the obtained results can be found at the following link: https://github.com/davidecopp/timetabling_dodm_2023.

2 Problem

The problem consists in scheduling a set E of exams during an examination period at the end of a semester. The examination period is divided into T ordered time-slots. A set S of student is considered. Each student is enrolled in a non-empty subset of exams.
Given two exams $e_1, e_2 \in E$, the quantity n_{e_1,e_2} represents the number of students enrolled in both of them. Two exams $e_1, e_2 \in E$ are said to be *conflicting* if they have at least one student enrolled in both of them, which means that $n_{e_1,e_2} > 0$.
Rules and regulations impose that conflicting exams cannot take place in the same time-slot. Moreover, to promote the creation of timetables more sustainable for the students, a penalty is assigned for each pair of conflicting exams scheduled up to a distance of 5 time-slots. In other words, given two exams $e_1, e_2 \in E$ scheduled at distance of i time-slots, with $1 \leq i \leq 5$, the associated penalty is given by:

$$2^{5-i}\frac{n_{e_1,e_2}}{|S|}. \tag{1}$$

The goal of Examination Timetabling Problem is to schedule the required exams into the available time-slots, taking into account and respecting to the following requirements:

1. Each exam is scheduled exactly once during the examination period;
2. Two conflicting exams cannot be scheduled in the same time-slot;
3. The total penalty resulting from the created timetable is minimized.

3 Model

In order to model the Examination Timetabling Problem, it is necessary to consider specific sets and matrices, for representing the available data. These ingredients are constructed in a generic way, and they are successively filled with the data associated to every available instance.
The necessary elements are:

- Set of the students involved in the creation of the examination timetable:

$$S = \{s_1, s_2, \ldots, s_{|S|}\}$$

- Set of the exams to be scheduled:

$$E = \{e_1, e_2, \ldots, e_{|E|}\}$$

- Set of the available time-slots:

$$T = \{t_1, t_2, \ldots, t_{|T|}\}$$

- Enrol matrix A: an $|S| \times |E|$ Boolean matrix, representing the exams to which each student is enrolled. Students are represented as rows of the matrix, while in the columns we have the available exams. Each value of the matrix is 1 if the considered student is enrolled in the considered exam, 0 otherwise. More precisely,

$$a_{s,e} = \begin{cases} 1 & \text{if student } s \text{ is enrolled in exam } e \\ 0 & \text{otherwise} \end{cases} . \tag{2}$$

- Conflict matrix C: an $|E| \times |E|$ integer matrix, representing the number of students enrolled in each couple of exams, and so the conflicting exams. Both rows and columns are filled with the set of available exams. Each value $c_{i,j}$ of the matrix represents the number of students that are enrolled in both the exams i and j. In particular,

$$c_{i,j} = n_{i,j},$$

  where $c_{i,j} > 0$ if and only if exam i and exam j are conflicting.
  Since C is symmetric by definition, in the implementation it has been set to be an upper triangular matrix, in order to avoid redundancy.

3.1 Variables

Since the aim is to assign exams to time-slots, the most intuitive way to represent this relationship is to create binary variables. Each variable is associated to a couple (e, t), where e is an exam and t is a time-slot, selected from sets E and T, respectively. The total number of variables is $|E| \times |T|$. The value of the variable is 1 if the considered exam is placed in the considered time-slot, and 0 otherwise. More precisely,

$$x_{e,t} = \begin{cases} 1 & \text{if exam } e \text{ is scheduled in time-slot } t \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

3.2 Constraints

The Examination Timetabling Problem requires to follow some rules for the creation of the exams' timetable. The following two constraints reflect the requirements for the formulation of the model relative to the basic problem.

- Each exam must be scheduled exactly once: for each exam e in set E, the sum of the variables $x_{e,t}$ for all the time-slots available in the problem should be equal to one. This means assigning, for each exam $e \in E$, just one among all the available time-slots (i.e. exactly one of the variables associated to e should be 1, while the others should be set to 0). In mathematical language, this constraint can be expressed by the following equation:

$$\sum_{t \in T} x_{e,t} = 1 \qquad \forall e \in E \tag{4}$$

- Conflicting exams cannot be scheduled in the same time-slot: the formulation of the problem impose that, if two exams are conflicting, they must be placed in different time-slots, in order to make it possible to students to attend both of them. As a consequence, the following constraint (applied if and only if the two exams are conflicting) is imposed in the formulation of the model:

$$x_{e_i,t} + x_{e_j,t} \leq 1 \quad \forall t \in T,\ \forall e_i, e_j \in E\ s.t.\ c_{e_i,e_j} > 0. \tag{5}$$

  According to these constraints, if e_1 and e_2 are two conflicting exams, every time exam e_1 is assigned to time-slot t (i.e. $x_{e_1,t} = 1$), the only possibility is to impose $x_{e_2,t} = 0$ and assign e_2 to another time-slot.
  On the other hand, if $x_{e_1,t} = 0$, $x_{e_2,t}$ could either be 0 or 1.This is because, if exam e_1 is not scheduled in time-slot t, then e_2 can be placed either in t or in any other time-slot, arbitrarily.
  The constraint is applied to every possible couple of exams and, for each couple, it is evaluated for all the time-slots in T.

3.3 Objective Function

The aim of the optimization problem is to minimize the total penalty of the created timetable. For this reason, the objective function of the model is the timetable's total penalty.
The objective function is intuitively computed as the sum of each couple of exams' penalty, after assigning the exams to the available time-slots and without violating any implemented constraint.
Given the penalty of a couple of exams, expressed by Equation (1), the following formula allows to compute the total penalty of a timetable:

$$Obj = \sum_{e_i,e_j \in E} \sum_{t_m,t_n \in T: 1 \leq |t_m - t_n| \leq 5} 2^{5-|t_m-t_n|} \frac{c_{e_i,e_j}}{|S|} x_{e_i,t_m} x_{e_j,t_n}. \tag{6}$$

Only conflicting exams ($c_{e_i,e_j} > 0$), scheduled in time-slots with a distance between 1 and 5, give a contribution to the total penalty.

In the second part of the equation, the variables x_{e_i,t_m} and x_{e_j,t_n} are added, in order to consider only the penalty of the couples of exams that are effectively present in the solution of the problem. In fact, if at least one of the two exams has not been scheduled in the considered time-slots, then at least one of the two variables is 0, resulting in no contribution to the objective function.

4 Equity measure

In addiction to the minimization of the total penalty of the timetable, there are many other different ways of measuring the goodness of a timetable: the equity measures. The idea is to substitute the original objective function of the basic model with a different equity measure, in order to evaluate and compare the results in the two cases.
An example of equity measure is the total number of times students have back-to-back exams. A back-to-back situation whenever a student is enrolled in exams scheduled in two consecutive time-slots. Obviously, a timetable with a lower number of back-to-back situations is more equitable for students with respect to one with an higher number of back-to-backs. This is due to the fact that, in the first case, students will have a lot more time to study or rest between exams, and thus the timetable will be more equal towards them.
As you can notice, we decided to count not the students having back to back exams (as it was suggested in the project description), but to take into account the total number of back-to-back situations. This decision has been made since we believed that the chosen measure could be better. For instance, if you consider a timetable where 5 students have back-to-back exams, it could appear apparently very good. The fact is that, if the those 5 students having back-to-back exams have not just a pair of conflicting exams which are scheduled in consecutive time-slots, but this event happens frequently inside the created timetable, the solution found really penalizes those 5 students.
The total number of back-to-back students is given by the formula:

$$b2b = \sum_{t=1}^{|T|-1} \sum_{e_i,e_j \in E} c_{e_i,e_j}(x_{e_i,t}x_{e_j,t+1} + x_{e_j,t}x_{e_i,t+1}), \tag{7}$$

where the terms inside the sum give a contribution to the total number of back-to-backs only if $c_{e_i,e_j} > 0$ (i.e. the two exams are conflicting) and $x_{e_i,t}x_{e_j,t+1} = 1$ or $x_{e_j,t}x_{e_i,t+1} = 1$(i.e. the solver effectively schedules the two exams in consecutive time-slots, either e_1 before e_2, or vice versa). This sum is evaluated $\forall t \in \{1, \ldots, |T| - 1\}$ and for all the couples (e_1, e_2).
This means that, every time the model schedules two exams in consecutive time slots, you want to count the number of students that are enrolled in both (adding them to the count of the back-to-backs). For this reason, the quantity $b2b$ is computed as a sum, with respect to all the possible couples of exams and to time. The terms of the sum are found as the product between the number of students that are enrolled in both the considered exams (different from 0 only if the two exams are conflicting) and the x's variables.
The sum $x_{e_i,t}x_{e_j,t+1} + x_{e_j,t}x_{e_i,t+1}$ is essential as the conflict matrix is upper triangular. In fact, we have to consider both the case in which e_i is scheduled before e_j and the opposite case, in which e_j is scheduled before e_i. Since the terms c_{e_j,e_i} are 0 for the area of the matrix

under the diagonal, we have to compute this sum, and multiply both terms by the value c_{e_i,e_j}, that is the same in both cases (we just change the order, but the considered exams are the same).

5 Additional restrictions

The basic model can be changed and updated by adding some additional constraints, which impose different rules for the creation of the timetable. In this case, the considered restrictions are the following:

- Change the constraints that impose that no conflicting exams can be scheduled in the same time slot. Instead, impose that at most 3 conflicting pairs can be scheduled in the same time slot.
  The idea of this restriction is to substitute the basic constraints that prevent the insertion of conflicting exams in the same time-slot with the creation of a completely different rule: there can be at most 3 conflicting pairs in the same time-slot. The formulation is the following:

$$\sum_{e_i,e_j \in E: c_{e_i,e_j} > 0} x_{e_i,t} x_{e_j,t} \leq 3 \quad \forall t \in T. \tag{8}$$

  In simpler words, for each time-slot $t \in T$, the couples of conflicting exams (i.e. the pairs $e_i, e_j \in E$ such that $c_{e_i,e_j} > 0$) have to be considered. Then, it is necessary to compute the product of the associated variables $x_{e_i,t}$ and $x_{e_j,t}$, in order to take into account only the couples where both the exams have been scheduled in the considered time-slot. Finally, a sum is made, with respect to the pairs of exams and this sum must be lower than or equal to three, for each time-slot in the timetable.

- At most 3 consecutive time slots can have conflicting exams.
  It is not so simple to add this condition: it is necessary to add additional binary variables z_t. These variables are defined as follows:

$$z_t = \begin{cases} 1 & \text{if time-slot } t \text{ contains at least a pair of conflicting exams} \\ 0 & \text{otherwise} \end{cases} . \tag{9}$$

  Since the new variables depend on the x's variables, they need to be defined trough a set of constraints:

$$\sum_{e_i,e_j \in E: c_{e_i,e_j} > 0} x_{e_i,t} x_{e_j,t} \leq M z_t \quad \forall t \in T, \tag{10}$$

$$z_t \leq \sum_{e_i,e_j \in E: c_{e_i,e_j} > 0} x_{e_i,t} x_{e_j,t} \quad \forall t \in T, \tag{11}$$

  where M is a large enough number (it has been set to 1000).
  The first inequality imposes that z_t is equal to 1 every time the left-hand side is greater than 1 (which means that these is at least one conflicting couple in time-slot t). On the other hand, the second inequality forces the value of z_t to 0 whenever the right-hand

side is 0 too (every time there are no couples of conflicting exams in time-slot t).
Thanks to this new variables, now the constraints can be modelled as follows:

$$5 - \sum_{i=0}^{2} z_{t+i} \geq 3z_{t+3} \quad \forall t \in \{1, ..., |T| - 3\}, \tag{12}$$

$$5 - \sum_{i=0}^{2} z_{t+i} \geq 3z_{t-1} \quad \forall t \in \{2, ..., |T|\}. \tag{13}$$

Every time 3 consecutive time-slots have conflicting exams, two restrictions are imposed: one for limiting the presence of conflicting exams in the following time-slot and the other to limit the presence of conflicting exams in the previous time-slot.
The two inequalities are substantially the same, except for the x variable in the right-hand side. The idea behind them is that, every time the z variable is equal to 1 in 3 consecutive time-slots (the sum of the z_{t+i} variables, for i from 0 to 2, is equal to 3), the variables z_{t+3} and z_{t-1} must be 0. Instead, if the sum is lower than 3, then z_{t+3} and z_{t-1} can assume value 0 or 1, indifferently.

- If two consecutive time slots contain conflicting exams, then no conflicting exam can be scheduled in the next 3 time slots.
  First of all, for creating this new constraint, it is necessary to impose an if condition which reflects the if-clause of the requirement: the additional rule need to be satisfied only if conflicting exams are contained in two consecutive time-slots of the created timetable. In order to respond to this necessity, another binary variable is added. This variable, $y_{e_i,e_j,t}$ is defined as following:

$$y_{e_i,e_j,t} = \begin{cases} 1 & \text{if } e_i \text{ and } e_j \text{ are scheduled in } t \text{ and } t+1, \text{ and } c_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{14}$$

  The values of y_t are imposed with this constraint:

$$y_{e_i,e_j,t} = x_{e_i,t}x_{e_j,t+1} + x_{e_j,t}x_{e_i,t+1}, \tag{15}$$

  which guarantees that $y_{e_i,e_j,t}$ is 0 if the right-hand size sum is 0. On the contrary, if the sum is 1, then $y_{e_i,e_j,t}$ is 1, too. There are no other possibilities, since the sum cannot take other values different from 0 and 1 (it cannot be 2, as each exam must be scheduled exactly once).
  If the above properties hold, then the following expression represents the restriction that needs to be imposed on the three consecutive time-slots:

$$\sum_{e \in E^*} \sum_{k=t+2}^{t+5} x_{e,k} y_{e_1,e_2,t} = 0, \tag{16}$$

  where $E^* = \{e \in E : e \neq e_i \text{ and } e \neq e_j \text{ and } (c_{e,e_i} > 0 \text{ or } c_{e,e_j} > 0)\}$ and $t \in \{1, \ldots, |T| - 5\}, \forall e_i, e_j \in E$.

The new set, E^*, allows to select all the exams that are different to the ones present in t and $t+1$ (e_i and e_j), and that are also in conflict with at least one exam between e_i and e_j. For all the exams in E^*, and for all the time-slots from $t+2$ to $t+5$, the idea is to compute the sum of the products between the variables $x_{e,k}$ and $y_{e_i,e_j,t}$ and force it equal to 0 (as no exam can be scheduled in the next 3 time slots). At this point, the if-clause of the requirement is guaranteed by the product with $y_{e_i,e_j,t}$.
The condition must hold $\forall t \in \{1, \ldots, |T| - 5\}$.

- Include a bonus profit each time no conflicting exams are scheduled for 6 consecutive time slots. Exploiting the already created z_t variables, this additional constraint can be computed as:

$$b_t = 1 - min\left(1, \sum_{i=0}^{5} z_{t+i}\right) \qquad \forall t \in \{1, ..., |T| - 5\}, \tag{17}$$

where b_t is a variable which is equal to 1 if the bonus is assigned to time-slot t and 0 otherwise. The variable b_t is activated (equal to 1) only if there are no conflicting exams scheduled for 6 consecutive time-slots (i.e. if the sum of the variables z_{t+i} is equal to 0). On the contrary, if the sum takes any other value different from 0, then $b_t = 1 - 1 = 0$, so the bonus will not be activated.
Due to time problems, we just wrote the idea of this additional constraint, without actually implementing it in Python. The idea is that, after defining it, b_t should be integrated with the objective function, in order to take into account also the bonus in the creation of the timetable, preferring solutions with an higher bonus.

6 Results

In this section, the results obtained with the basic model, after imposing a time-limit of 1000 seconds on the solver (except for instance 11, which required more time), compared with the established benchmarks (best solution available for each instance), are shown.

Looking at Table 1, it is possible to make a comparison between the results obtained with the implementation of the presented model and the benchmark solutions.
It can be immediately noticed that the model works perfectly with the test instance: in this case the value of the objective function is equal to 3.375 in both the columns. Therefore, the result suggests that the model works well, finding the optimal solution in just a couple of seconds.
On the contrary, with all the other instances the situation is not the same. In general, as expected, the results found are greater than the benchmarks, since the latter represent the best solution available for each instance. For some instances, the value is very close to the benchmark, especially for the first instance, but also for instance 7. For the others, we found feasible solutions that are not so close to the benchmark, probably due to the fact that those results have been obtained using more sophisticated methods and required hours of computation. At least, we found feasible solutions for each instance, that, in some cases, are not even so far from the benchmark.

Name	Result	Benchmark
test	3.375	3.375
instance01	157.357	157.033
instance02	42.527	34.709
instance03	46.338	32.627
instance04	14.223	7.717
instance05	18.945	12.901
instance06	6.535	3.045
instance07	11.492	10.050
instance08	27.597	24.769
instance09	16.429	9.818
instance10	8.883	3.707
instance11	9.657	4.395

Table 1: Results and benchmarks for the provided instances.

6.1 Equity measure results

Here, we show the results obtained from the implementation of the basic model, with the substitution of the penalty objective function with the back-to-back objective function. The time limit was still fixed to 1000 seconds.

Name	Back-to-backs	Penalty
test	0	3.375
instance01	3021	163.358
instance02	1315	48.358
instance03	1208	46.925

Table 2: Comparison between back-to-backs and penalty.

Thanks to this table, it is possible to see the obtained values of objective function for some of the provided instances. Moreover, in the third column of the table, you can see the values obtained by taking the solutions found and computing their correspondent penalty.
The two columns allow to make a comparison between the results obtained with the two objective functions. As you can notice, in the test instance there are 0 back-to-backs, meaning that there are no conflicting exams scheduled in consecutive time-slots.
For the first instance, the best solution found contains 3021 back-to-backs, which means that, since there are 611 students, each of them has to deal on average with almost 5 back-to-back situations. This makes sense, as the instance contains 139 exams to be scheduled in just 13 time-slots (and each student is enrolled to many exams, in various cases even 11). So, it seems pretty realistic that the number of back-to-backs is so high. Moreover, the obtained solution returns a penalty value of 163.358, meaning that minimizing the number of back-to-backs doesn't bring to the same solution reached with the penalty objective function. This result seems fair since the two objective functions reflect different purposes.
For instances 2 and 3, the back-to-back values are smaller. In particular, in the third instance,

the solution gives a penalty of 46.925, whose value is very close to the one obtained with the previous objective function, thus suggesting that in this case, minimizing back-to-backs almost equilizes the performance of minimizing the total penalty of the timetable.

6.2 Additional restrictions results

The following table represents the results for the test and for instance 01 achieved in the model composed of the original penalty objective function, plus the additional restrictions already described in section 5.

Name	Additional	Basic
test	3.375	3.375
instance01	193.817	157.357

Table 3: Penalty computed with additional restrictions.

For the other instances, due to the big number of constraints to check, to the huge dimension of the instances themselves, and to the time limit of 1000 seconds, the solver wasn't able to find feasible solutions. It doesn't mean that the involved instances are unfeasible, but simply that a solution cannot be found in such a limited time.

7 Procedure

In this section the procedure in order to run the code is provided. Inside the GitHub repository (https://github.com/davidecopp/timetabling_dodm_2023) it is possible to find the following files:

- main.py;
- instances;
- solutions;
- utils.

To run the optimization timetabling problem for a single instance, the command to be launched is the following:

```
python main.py [instance] [objective function] [model type]
```

In particular:

- *instance* should be passed as **instanceXX**;
- *objective function* can take **penalty** or **b2b** value;
- *model type* can take **base** or **advanced** value.