Discrete Optimization and Decision Making - Final Project
Examination Timetabling Problem

Francesco Mengalli - VR480249

Academic year 2022/2023

1 Introduction

The aim of this report is to illustrate the solution for the Examination Timetabling Problem. In particular, this project work contains a formulation of this problem as an Integer Linear Programming model and a presentation, followed by an explanation, of the variables involved, the constraints, and the objective function. This specific implementation uses Python 3.9.7 with the last release of Gurobi Optimizer package available at this date.

Following a presentation of the Examination Timetabling Problem, underlining all the requirements, there is a description of the model with further explanation of the choices made. Below, two variants of the first basic problem are shown: on one hand, the original objective function is replaced by a new equity measure, while in the second one, there are new constraints with more restrictions to satisfy. In the end, the model's results are shown concerning a set of provided instances. The results are also compared to already available benchmarks.

The implementation of this problem in Python code, the test instances, and the obtained results, can be found at the following link:
`https://github.com/francescomengalli/timetabling_dodm_2023/tree/main`.

2 The problem

The problem involves 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 students is considered, where 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, meaning that $n_{e_1,e_2} > 0$.

Rules and regulations dictate that conflicting exams cannot take place in the same time-slot. Additionally, to encourage the creation of timetables more sustainable for students, a penalty is assigned for each pair of conflicting exams scheduled up to a distance of five time-slots. In other words, given two exams $e_1, e_2 \in E$ scheduled at a distance of i time-slots, with $1 \leq i \leq 5$, the associated penalty is given by:

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

The aim of the Examination Timetabling Problem is to schedule the required exams into the available time-slots, considering and respecting 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 The model

To model the Examination Timetabling Problem, specific sets and matrices must be considered to represent the available data. These components are constructed in a generic manner and are successively filled with the data associated with every available instance.

The necessary elements are:

- Set of students involved in creating the examination timetable:

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

- Set of exams to be scheduled:

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

- Set of 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 columns represent available exams. The entries of this matrix take binary values, more precisely,

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

- Conflict matrix C: an $|E| \times |E|$ boolean matrix representing the number of students enrolled in each pair of exams, indicating conflicting exams. Rows and columns are filled with the set of available exams. Each value $c_{i,j}$ of the matrix represents the number of students enrolled in both 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, it has been implemented as an upper triangular matrix to avoid redundancy and to improve the efficiency of the code itself.

3.1 Variables

As the goal is assigning exams to time-slots, the most intuitive way to represent this relationship is by creating binary variables. Each variable is associated with a pair (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 $|E| \times |T|$. More precisely,

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

3.2 Constraints

The Examination Timetabling Problem requires to satisfy some conditions for the creation of the timetable. The following reflect the requirements for the formulation of the model concerning 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. So:

$$\sum_{t \in T} x_{e,t} = 1 \quad \forall e \in E$$

- **Conflicting exams cannot be scheduled in the same time-slot:** The formulation of the problem imposes that if two exams are conflicting, they must be placed in different time-slots to allow students to attend both of them. Consequently, the following constraint (applied if and only if the two exams are conflicting) is imposed in the formulation of the model:

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

According to these constraints, if e_i and e_j are two conflicting exams, every time exam e_i is assigned to time-slot t (i.e., $x_{e_i,t} = 1$), the only possibility is to impose $x_{e_j,t} = 0$ and assign e_j to another time-slot.

Viceversa, if $x_{e_i,t} = 0$, $x_{e_j,t}$ could either be 0 or 1. This is because if exam e_i is not scheduled in time-slot t, then e_j can be placed either in t or in any other time-slot, arbitrarily.

The constraint is applied to every possible pair of exams, and for each pair, and it is evaluated for all the time-slots in T.

3.3 Objective Function

Since the aim of the optimization problem is to minimize the total penalty of the created timetable, the objective function of the model is the timetable's total penalty. The objective function is intuitively computed as the sum of each pair of exams' penalty, after assigning the exams to the available time-slots and without violating any implemented constraint.

Given the penalty of a pair of exams explained above, the following formula allows computing the total penalty of a timetable:

$$\text{Obj} = \sum_{e_i,e_j} \sum_{1 \leq |t_m - t_n| \leq 5} 2^{5-|t_m-t_n|} \cdot \frac{c_{e_i,e_j}}{|S|} \cdot x_{e_i,t_m} \cdot x_{e_j,t_n}$$

$$\text{where } e_i, e_j \in E, i \neq j, \text{and } t_m, t_n \in T$$

It follows that only conflicting exams ($c_{e_i,e_j} > 0$), scheduled in time-slots with a distance between 1 and 5, contribute 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 pairs of exams that are effectively present in the solution of the problem. 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 addition to minimizing the total penalty of the timetable, there are many other ways of measuring the goodness of a timetable: equity measures. The idea is to substitute the original objective function of the basic model with a different equity measure to evaluate and compare the results in the two cases.

For example, is the total number of times students have back-to-back exams. A back-to-back situation occurs 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 compared to one with a higher number of back-to-backs. This is because, in the first case, students will have more time to study or rest between exams, making the timetable more equal towards them.

As you can notice, the decision made is to count not the students having back-to-back exams (as suggested in the project description), but to consider the total number of back-to-back situations. This choice has been made since the chosen measure could be better for the aim of this project. 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 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:

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

where the terms inside the sum contribute 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_i before e_j, or vice versa). This sum is evaluated for every $t \in \{1, \dots, |T| - 1\}$ and for all pairs (e_i, e_j).

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 possible pairs of exams and 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, there is to compute the sum, and multiply both terms by the value c_{e_i,e_j}, that is the same in both cases (there is only a change in the order).

5 Additional Restrictions

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

- **Change the constraints regarding conflicting exams in the same time slot:** Instead of preventing the insertion of conflicting exams in the same time-slot, impose that at most 3 conflicting pairs can be scheduled in the same time slot. The formulation is as follows:

$$\sum_{e_i,e_j \in E} x_{e_i,t}x_{e_j,t} \leq 3 \quad \forall t \in T$$

$$\text{where } c_{e_i,e_j} > 0$$

  In simpler words, for each time-slot $t \in T$, consider pairs of conflicting exams (i.e., pairs $e_i, e_j \in E$ such that $c_{e_i,e_j} > 0$). Then, compute the product of the associated variables $x_{e_i,t}$ and $x_{e_j,t}$ to take into account only the pairs where both exams have been scheduled in the considered time-slot. Finally, sum these products for all 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:** Adding this condition requires the introduction of additional binary variables z_t. These variables are defined as follows:

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

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

$$\sum_{e_i,e_j \in E} x_{e_i,t}x_{e_j,t} \leq M z_t \quad \forall t \in T$$

$$z_t \leq \sum_{e_i,e_j \in E} x_{e_i,t}x_{e_j,t} \quad \forall t \in T$$

  where $c_{e_i,e_j} > 0$ and M is a large enough number (in this project, 1000).

The first inequality ensures that z_t is equal to 1 whenever the left-hand side is greater than 1 (indicating at least one conflicting couple in time-slot t). 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 these new variables, the constraints can now be modeled as follows:

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

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

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 essentially the same, except for the x variable on 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:** To create this new constraint, it is necessary to impose an if condition which reflects the if-clause of the requirement. The additional rule needs 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 follows:

  $$y_{e_i,e_j,t} = \begin{cases} 1 & \text{if } e_i \text{ and } e_j \text{ are scheduled in } t \text{ and } t+1, \text{ and } c_{e_1,e_j} > 0 \\ 0 & \text{otherwise} \end{cases}$$

  The values of y_t are imposed with this constraint:

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

  This guarantees that $y_{e_i,e_j,t}$ is 0 if the right-hand side sum is 0. Otherwise, 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 values other than 0 and 1 (it can not 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:

  $$\sum_{e \in E^*} \sum_{k=t+2}^{t+5} x_{e,k} y_{e_i,e_j,t} = 0$$

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 $t \in \{1, \dots, |T| - 5\}$, $\forall e_i, e_j \in E$.

The new set, E^*, allows selecting all the exams that are different from 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 to be 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 $\forall t \in \{1, \dots, |T| - 5\}$.

- **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:

$$6 - \sum_{i=0}^{5} z_{t+i} \geq 6b_t \quad \forall t \in \{1, \dots, |T| - 5\}$$

$$b_t \geq 1 - \sum_{i=0}^{5} z_{t+i} \quad \forall t \in \{1, \dots, |T| - 5\}$$

where b_t is a variable which is equal to 1 if the bonus is assigned to time-slot t and 0 otherwise. After defining it, b_t should be integrated with the objective function. 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.

Because of time and hardware limitations, this aspect is not implemented in the Python code of this project. The idea is that, after defining it, b_t should be included in the objective function to consider the bonus in the creation of the timetable, preferring solutions with a higher bonus.

6 Results

In this section, there are the results obtained with the basic model, after imposing a time-limit of 1000 seconds on the solver, compared with the established benchmarks (best solution got for each instance).

Looking at the first table, 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, therefore, the result suggests that the model works well, finding the optimal solution in few seconds.

The situation changes with the other instances. In general, as expected, the results found are greater than the benchmarks, since the latter represent the best solution available for each instance. The table below presents these results

Name	**Result**	**Benchmark**
test	3.375	3.375
instance01	157.736	157.033
instance02	44.23	34.709
instance03	51.429	32.627
instance04	12.565	7.717
instance05	19.342	12.901
instance07	12.967	10.050
instance08	30.202	24.769
instance09	14.477	9.818

Feasible solutions were found for these instances, with results sometimes not so close to the benchmark. This discrepancy may be due to the use of more sophisticated methods and extended computation time. Nevertheless, feasible solutions were obtained for these instances, some of which are reasonably close to the benchmark. Other instances like 06, 10 ad 11 gave no feasible solution, probably this is due to the hardware processor used.

6.1 Equity Measure Results

A second table compares the results obtained with the basic model using the back-to-back objective function, replacing the penalty objective function.

Name	**Back-to-backs**	**Penalty**
test	0	3.375
instance01	3021	165.347
instance02	1315	49.357
instance03	1208	47.531

This table displays the obtained values of the objective function for almost all the instances. Additionally, the third column presents the penalty values obtained by taking the solutions found with the back-to-back objective function. The two columns allow a comparison between the results obtained with the two objective functions. For instance, in the test instance, there are 0 back-to-backs, indicating that there are no conflicting exams scheduled in consecutive time-slots. In 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 an, since each student is enrolled in to many exams).

Continuing the analysis, it appears realistic that the number of back-to-backs is high, especially given the constraints of scheduling exams within a limited number of time-slots, with each student enrolled in multiple exams. Therefore, the observed high back-to-back count seems plausible. Additionally, the solution obtained for the first instance returns a penalty value of 165.347. This implies that minimizing the number of back-to-backs doesn't lead to the same solution as minimizing the total penalty of the timetable. This result follows by the fact that the two objective functions serve different purposes.

For instances 02 and 03 the back-to-back values are smaller and the values returned by modified objective function are very close to the ones obtained with the previous objective function. This suggests that, in this case, minimizing back-to-backs almost equalizes the performance of minimizing the total penalty of the timetable.

6.2 Additional Restrictions Results

This last table shows the results for the test and instance 01 achieved in the model composed of the original penalty objective function, plus the additional restrictions described in the section above.

Name	**Additional Restrictions**	**Basic**
test	3.375	3.375
instance01	194.088	157.736

For the other instances, due to the large number of constraints to check, the huge dimensions of the instances, and the time limit of 1000 seconds, the solver wasn't able to find feasible solutions. This doesn't imply that the involved instances are unfeasible, but rather that a solution cannot be found with this conditions.

7 Procedure

In this section you can find the procedure to run this code. First of all, the code, where the following files can be found, is available in a GitHub repository at this link: `https://github.com/francescomengalli/timetabling_dodm_2023/tree/main`.

- `main.py`;
- `instances`;
- `solutions`.

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] [time]
```

- `[instanceXX]` should be passed as the specific instance, such as `instance01`
- `[objective function]` can take either `penalty` or `b2b` value
- `[model type]` can take either `base` or `advanced` value
- `[time]`: set the desired time in seconds, 1000 for the results shown on this report.