Journal of Scheduling manuscript No.
(will be inserted by the editor)

Disruptions in Timetables: A Case Study at Universidade de Lisboa

Alexandre Lemos · Pedro T. Monteiro · Inês Lynce

Received: date / Accepted: date

Abstract Solving university timetabling problems is a large and complex task. Moreover, every new academic year, a new timetable is likely to be scheduled due to minor disruptions. However, a timetable does not need to be periodically scheduled from scratch. Solving a timetabling problem from scratch can produce a completely different solution from the previous one, thus creating undesirable changes for many actors.

This paper addresses the Minimal Perturbation Problem (MPP) in university timetabling. Given a set of disruptions that make a timetable no longer feasible, a solution to the MPP is the *closest* new feasible timetable with respect to the original timetable. We propose and analyze two different integer programming models to encode the MPP. To validate the proposed models, disruptions are randomly generated based on the probability distributions learned from the history of timetables over the last five years in the Instituto Superior Técnico (IST) the engineering school from Universidade de Lisboa. Overall, our models, combined with an incremental approach, are shown to be able to efficiently solve all problem instances.

Keywords Minimal Perturbation Problem · University Timetabling · Disruption · Integer Programming.

This work was supported by Universidade de Lisboa, Instituto Superior Técnico and Departamento de Engenharia Informática (DEI) and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID multi-annual funding).

A. Lemos · P.T. Monteiro · I. Lynce
Instituto Superior Técnico, Universidade de Lisboa
INESC-ID, Rua Alves Redol, 9, 1000-029 Lisboa, Portugal
E-mail: {alexandre.lemos,pedro.tiago.monteiro,ines.lynce}
@tecnico.ulisboa.pt

1 Introduction

In this paper, we address modeling and reformulation of course timetabling problems in real-world scenarios, one of the major open challenges in university timetabling problems (McCollum 2006; Vrielink et al. 2019). University timetabling problems are known to be NP-complete (Even et al. 1976) given that the search space for possible solutions grows exponentially with the number of lectures. Formally, one can define course timetabling problems as follows:

Definition 1 (Course Timetabling) A solution to a course timetabling problem is an assignment to all lectures (L) to suitable rooms (R) and time slots (T) in such a way that all the constraints ($\mathcal{C}$) are satisfied.

Every semester a new timetable needs to be created. Usually, the number of changes between the timetables of consecutive years is small. These changes can be of different types: either constraints are added/changed, or the domain of a variable is changed. These changes may cause the original solution to be infeasible. Solving the problem from scratch may negatively affect many of the actors involved. However, it is not necessary to generate a completely new timetable each semester. This problem can be considered an Minimal Perturbation Problem (MPP) where the existing solution is no longer feasible and one needs to find the *closest* feasible solution. Formally,

Definition 2 (MPP) A solution to an MPP is a solution to a course timetabling problem subject to a new set of constraints and/or domain of variables while minimizing a distance function δ, which computes the distance between the original solution and the new feasible solution.

Solving the MPP instead of solving the problem from scratch can improve user satisfaction with the resulting timetable. Actually, this is the approach taken by the academic services at IST. Most often, the disruptions result from changes in teacher-lecture allocation, curricular plans and number of students. In addition, when specific problems referring to previous timetables are reported, one can learn from previous mistakes. Besides these changes, the academic services assume that everything else in the timetable is acceptable. For this reason, and if possible, that part of the timetable will continue unchanged. This assumption allows the services to reduce the scope of the problem.

In the context of university timetabling, there is one more scenario that is considered an MPP. The university is a dynamical system and thus the timetables should be able to change even during their execution.

Example 1 Let us consider the example shown in Figure 1a. The weekly timetable shows the different lectures of the different courses a student can attend. Each student having this curricular plan must attend some lectures from the four courses represented (A to D). For each course, a student must attend all theoretical lectures and only one practical / laboratory lecture. For example, a student must attend all A^{T} (theoretical) lectures in the schedule and only one of the two A^{L} (laboratory) possible lectures.

Consider the following disruptions applied to Example 1: (i) room L2 is closed for renovations for a long period of time and (ii) the number of teachers available to teach A^{L} is now only one. Consider that we want to cause the smallest number of perturbations (δ) to the original solution. Disruption (i) reduces the domain of the problem. In this small example, one lecture is assigned to L2. Therefore, the original solution (1a) is no longer feasible. If one considers that there are only two laboratories (L1 and L2), and that L1 is already taken in the required time slot, then we conclude that the optimal solution requires two perturbations to the original solution (change room and time). Disruption (ii) adds a new *no overlap* constraint to the model. Note, however, that any solution containing the perturbations resulting from the first disruption already works out for this disruption. A possible solution is shown in Figure 1b.

The contribution of this paper is two-fold: (i) two different integer programming models (Boolean and mixed) to encode the MPP applied to university timetabling, and (ii) an incremental algorithm to solve these models efficiently. We showcase the application of the models in real-world problem instances using data sets from IST. The disruptions are generated with different probabilities to match the history of disruptions of the case study.

This paper is organized as follows. Section 2 provides a brief overview of the relevant state of the art. Section 3 formally describes the problem. Section 4 describes the different models to encode the problem. Section 5 discusses the generation of disruptions based on history and shows the computational results for the different models and disruptions. Finally, Section 6 concludes the paper and addresses future work.

2 Related work

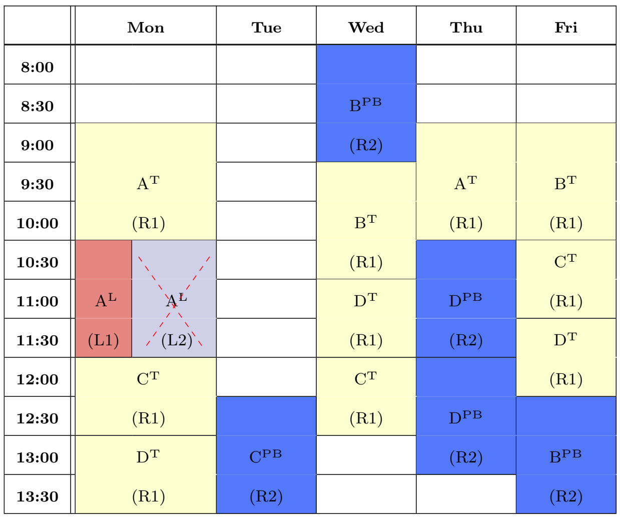
2.1 Timetables

University timetabling problems (Kingston 2013; McCollum 2006; Vrielink et al. 2019) can be classified into two major categories: examination timetabling (Müller 2009) and course timetabling problems (Di Gaspero et al. 2007). Many different approaches have been successfully applied for solving these problems, namely Constraint Programming (CP) (Müller et al. 2004), Answer Set Programming (ASP) (Banbara et al. 2019), Boolean Satisfiability (SAT) (Achá and Nieuwenhuis 2014), Integer Linear Programming (ILP) (Lemos et al. 2019; Lindahl et al. 2019; Lach and Lübbecke 2008; Vermuyten et al. 2016; Cacchiani et al. 2013), genetic algorithms (Pillay and Özcan 2019) and local search (Lemos et al. 2019; de Souza Rocha et al. 2012; Bellio et al. 2012).

In recent years, a siginificant improvement in solving course timetabling problems have been achieved (Bettinelli et al. 2015). Behind this progress lies the public data sets from ITC (Di Gaspero et al. 2007), which are a simplified version of the real timetabling problem at the University of Udine. For this reason, the progress made still presents a gap between theory and practice (McCollum 2006) since it does not capture the full complexity of the real world problem.

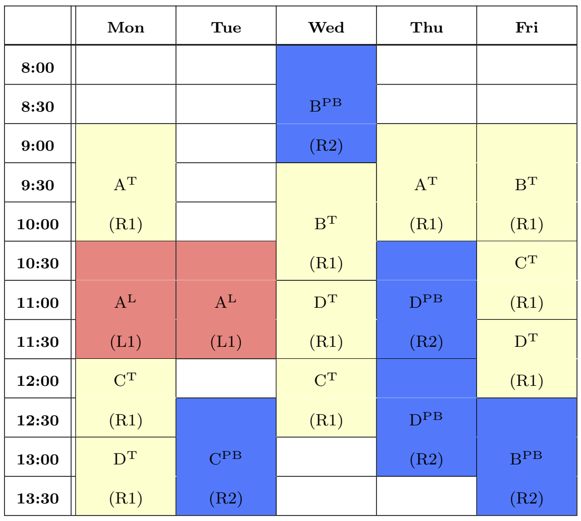
Vermuyten et al. (2016) proposed a two-stage approach to optimize student flows using ILP. The methods were tested using the real data of Katholieke Universiteit Leuven (KUL). The KUL data is more complex than the simplified ITC data. However, it also adds specific problems that are inherent to the KUL data.

Lemos et al. (2019) proposed three different approaches to optimize space usage in university timetabling while ensuring the maximum number of students are able to attend their lectures. Two of these approaches solve the problem using greedy heuristics which are able to solve efficiently the problem. Nevertheless, they fall short of the optimal value. Therefore, the authors proposed an ILP approach which is slower



	Mon	Tue	Wed	Thu	Fri
8:00					
8:30			B^{PB}		
9:00			(R2)		
9:30	A^{T}			A^{T}	B^{T}
10:00	(R1)		B^{T}	(R1)	(R1)
10:30			(R1)		C^{T}
11:00	A^{L} A^{L}		D^{T}	D^{PB}	(R1)
11:30	(L1) (L2)		(R1)	(R2)	D^{T}
12:00	C^{T}		C^{T}		(R1)
12:30	(R1)		(R1)	D^{PB}	
13:00	D^{T}	C^{PB}		(R2)	B^{PB}
13:30	(R1)	(R2)			(R2)

(a) Before



	Mon	Tue	Wed	Thu	Fri
8:00					
8:30			B^{PB}		
9:00			(R2)		
9:30	A^{T}			A^{T}	B^{T}
10:00	(R1)		B^{T}	(R1)	(R1)
10:30			(R1)		C^{T}
11:00	A^{L}	A^{L}	D^{T}	D^{PB}	(R1)
11:30	(L1)	(L1)	(R1)	(R2)	D^{T}
12:00	C^{T}		C^{T}		(R1)
12:30	(R1)		(R1)	D^{PB}	
13:00	D^{T}	C^{PB}		(R2)	B^{PB}
13:30	(R1)	(R2)			(R2)

(b) After

Fig. 1: Timetable for a class of students (a) before and (b) after occurring two disruptions: (i) an *unavailability* constraint over room L2 and (ii) a *no overlap* constraint *w.r.t.* the two lab lectures of course A. The colors represent the different rooms the lectures are assigned to.

but it is able to find the optimal solution for every instance tested. All aproaches were tested using a very large data set corresponding to the data from IST. The work proposed in this paper is an extension to our previous work (Lemos et al. 2019) in order to solve MPP.

2.2 Minimal Perturbation Problem

Given a solution to a problem instance and a set of disruptions which make the solution no longer feasible, solving the MPP is the task of finding a new feasible solution requiring the smallest number of perturbations (Sakkout and Wallace 2000; Zivan et al. 2011; Phillips et al. 2017). The MPP adds an optimization criterion to the course timetabling problem. The goal of this criterion is to evaluate the distance between the solutions, *i.e.* the number of perturbations required to go from one solution to another (formally defined in Section 3). The most commom approach (Zivan et al. 2011; Phillips et al. 2017; Müller et al. 2004; Banbara et al. 2019; Lindahl et al. 2019) is to apply the Hamming Distance (Hamming 1950). However, this metric lacks domain knowledge.

Müller et al. (2004) proposed the iterative forward search algorithm to solve the MPP applied to university timetabling. Since this method is a local search method, it does not ensure completeness. Phillips et al. (2017) use integer programming to solve MPP on instances from the University of Auckland. The proposed method tries to solve the problem in the smallest possible neighborhood. If feasible solution is not found, then the neighborhood is gradually expanded until either a feasible solution is found or the neighborhood includes the whole search space.

Banbara et al. (2019) proposed an ASP based tool to compute Pareto fronts with two objectives: (i) minimize the number of soft constraints unsatisfied and (ii) minimize the number of perturbations. The tool solves the MPP if and only if the disruption changes the course timetabling problem by adding or removing constraints. Therefore, all disruptions that cause the domain of the problem to change cannot be solved by this tool.

Recently, Lindahl et al. (2019) proposed a bi-objective integer programming model to solve MPP applied to curriculum-based course timetabling. The goal is to find Pareto-optimal solutions for these two objectives: (i) minimize the number of perturbations and (ii) minimize the number of soft constraints unsatisfied. The results show that the MPP solutions often have low quality and that allowing few more perturbations can significantly improve the quality. Lindahl et al. (2019) approach evaluated with data sets from the 2007 International Timetabling Competition (ITC).

2.3 Disruptions in University Timetabling

There are different types of disruptions which can be categorized as follows: changes in the domain ($\mathcal{D}$) of the sets (*e.g.* remove a room) or a change in the set of constraints ($\mathcal{C}$) (*e.g.* remove a room for a day). Table 1 summarizes the different disruptions that can occur in

Table 1: Summary and literature review of the most common disruptions in a university scenario. The implications in problem structure are classified as changes in the domain ($\mathcal{D}$) of the sets or changes in the set of constraints ($\mathcal{C}$).

Disruption	Lindahl et al. (2019)	Phillips et al. (2017)	Müller et al. (2004)	Banbara et al. (2019)	Other	Cause
Room Stability				✓		$\mathcal{C}$
Overlap/No Overlap					✓	$\mathcal{C}$
Invalid Assignment	✓		✓			$\mathcal{C}$
Invalid/Preference Time	✓	✓				$\mathcal{C}$
Invalid/Preference Room					✓	$\mathcal{C}$
Remove Room for a day	✓					$\mathcal{C}$
Remove Room		✓				$\mathcal{D}$
Insert Curriculum	✓					$\mathcal{D}$
Modify Enrollments		✓				$\mathcal{D}$
Modify # Lectures					✓	$\mathcal{D}$

university timetabling. The most common disruptions that can occur when re-solving university timetabling problems are following:

– *Room stability*: disruption forces all lectures of a course to be scheduled in the same room.
– *Overlap* (*no overlap*): disruption refers to the addition of a constraint to forbid (force) lectures to be taught simultaneously. For example, *no overlap* disruption may arise when the number of teachers assigned to a course reduces to one.
– *Invalid assignment*: disruption refers to a problem in the assignment of the lecture to a room or to a time slot. This disruption abstracts the origin of the problem (time or room assignment). However, in a real-world scenario it is more common to find more specific disruptions.
– *Invalid time* and *invalid room*: are disruptions that make an assignment of a lecture to a time slot and a room invalid, respectively. These two last disruptions are more specific than the *invalid assignment* disruption. Note that, these disruptions can be further specified when only a time slot or a room is unavailable to only a specific set of courses. *Preference time* and *room preference* are the opposite disruptions of the *invalid time* and *invalid room.*
– *Remove room for a day* and *remove room*: cause a room unavailability for an assignment for a day and forever, respectively.
– *Insert curriculum*: adds a new set of courses. Each lecture of the new courses cannot be overlapped in time. All constraints relating to lectures are applied (*e.g.* room capacity).
– *Modify enrollments*: modifies the number of students that are enrolled in a lecture, since it naturally between the execution of consecutive years. The disruption may not require changes in the original solution since the difference may still be supported.
– *Modify Number of Lectures*: adds/removes the number of possible lectures that a student can attend. The number of lectures does not change for each student. However, the students have more/less flexibility in their choice of timetable. This disruption can be a side effect of the *modify enrollments* disruption.

The model proposed by Lindahl et al. (2019) only considered the following disruptions: *remove room for a day*, *invalid time*, *insert curriculum*, and *invalid assignment.* With these set of disruptions, one can define an upper bound on the number of satisfied soft constraints since they cannot improve the quality of the original solution. The disruptions reduce the search space. However, in this work, we consider different types of disruptions, such as disruptions that change the problem variable's domain and may lead to a solution with better quality. Therefore, we cannot simply apply the algorithm proposed by Lindahl et al. (2019) as shown in Example 2.

Example 2 Let us consider that the lecture C^{T}, shown in Figure 1a, has 125 students enrolled. Room R1 has a capacity of only 110 students. Consider that R1 is the only room with enough capacity to fit more than 100 students. When generating the new timetable (based on the last semester), a disruption causes the number of students enrolled in C^{T} to be reduced to only 105. This disruption causes the overall quality of the timetable to improve.

3 Problem Definition

In this section we formally introduce the problem and the notation used throughout the text.

3.1 Preliminaries

Let us consider a set of periods $P \in \{0, ..., 120\}$ corresponding to all possible time slots of a working week ($|P| = |D| \times |T|$). These periods are separated in working days $D \in \{0, ..., 4\}$ (0 corresponding to Monday, 1 to Tuesday, and so on). Each day has a set of consecutive working time slots of half an hour, $T \in \{0, ..., 23\}$[1]. The university has a set R of rooms where lectures can be scheduled. All university lectures L (from different courses) have to be assigned to a time slot and to a room.

Consider a set of courses C, with each course $c \in C$ having a set of lectures $\mathcal{L}_c$ in which a set of students (S) can be enrolled in. Each set $\mathcal{L}_c$ is composed by n disjoint subsets of lectures ($L_c^{1,...,n} \subseteq \mathcal{L}_c$) of different types (*e.g.* theoretical, practical, laboratory). A student enrolled in course c must attend exactly one lecture of each set L_c^i. Each subset of lectures $L_c^i \subseteq \mathcal{L}_c$, where $1 \leq i \leq n$, has a value $overlap_{L_c^i}$ associated, with $0 < overlap_{L_c^i} \leq |L_c^i|$, where $overlap_{L_c^i}$ represents the number of lectures of L_c^i that can be overlapped. In other words, $overlap_{L_c^i}$ represents the smallest number of teachers[2] in charge of lectures in L_c^i.

Furthermore, a lecture $l \in L$ is characterized by: a set of enrolled students ($S_l \subseteq S$) of size std_l; a set of suitable rooms ($R_l \subseteq R$); a set of suitable days ($D_l \subseteq D$); a set of suitable time slots ($T_l \subseteq T$); and its duration ($len_l > 1$).

Each student $s \in S$ has a set of lectures $L_s \subseteq L$ where he is enrolled in. Since all weeks have basically the same lectures, one can generate the timetables for one week and generalize for the weeks of the whole semester.

Finally, each lecture has to be assigned to a suitable room (R_l). Each room r, with $r \in R$, has an ideal capacity, $cap_r > 0$. To ensure a fair distribution for students, we add a slack variable of α, representing the percentage of students enrolled in a lecture for which it is acceptable to be over the ideal capacity of the assigned room (*i.e.* overbooking).

The concepts described in this section can be converted to decision variables. However, the definition of these variables depends on the model (see Section 4).

[1] We consider that the rooms are available for use only between 8am and 8pm, corresponding to a total of 12 hours and 24 slots of half an hour.

[2] The assignment of teachers to lectures is only performed after the schedule is created. Therefore, this number is computed based on the timetables from the previous year.

3.2 Constraints

The set of constraints ($\mathcal{C}$) considered in this work are as follows:

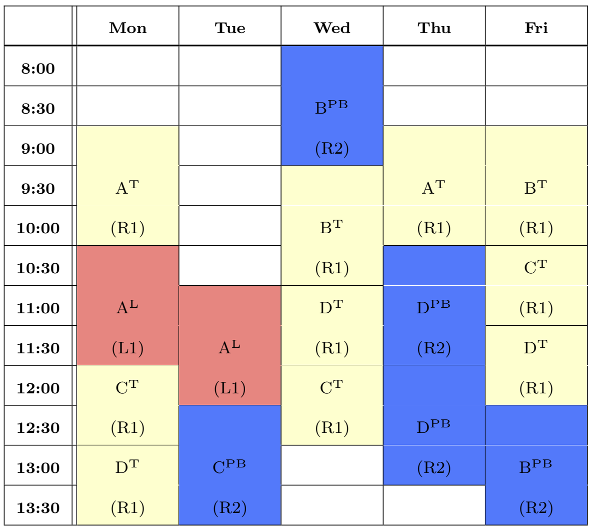
1. *Lectures to time slots*: All lectures must be assigned to the corresponding time slots.
2. *Consecutive time*: A lecture must be taught in consecutive time slots on the same day.
3. *Lectures to rooms*: All lectures that require room assignment ($R_l \neq \emptyset$) must be assigned to exactly one room.
4. *Student's conflicts*: All students must be able to attend the lectures for which they are enrolled. This way one can avoid some complex curricular rules since the students represent a specific path in the curricular plan. Otherwise, for each course, it would be required to encode multiple combinations of lectures that a student can attend.
5. *Room conflicts*: A room can have at most one lecture scheduled per time slot per day.
6. *Teacher conflicts*: At-most $overlap_{L_c^i}$ lectures from the same course can be lectured at the same time.
7. *Capacity*: Ensures that, in the worst case, only α% of the students enrolled may not be seated.
8. *Invalid Room* (r, l): The assignment of the lecture l to the room r is invalid.
9. *Invalid Time* (d, t, l): The assignment of the lecture l to the slot t in day d is invalid.
10. *Overlap* $(l1, l2)$: The lectures $l1$ and $l2$ have to be assigned to the same time slot of the same day.
11. *Remove room* (r): The room r cannot be used. This constraint can be easily generalized to remove room r for short period of time.

These constraints were provided by domain experts of the university academic services.

3.3 Optimization Criteria

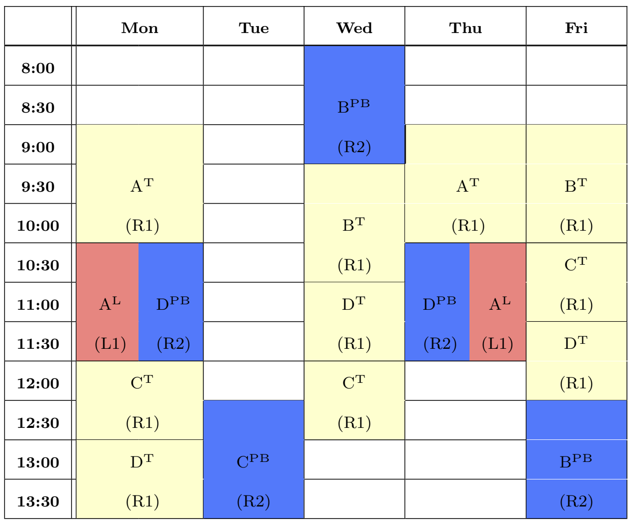
In addition to the different constraints, one needs to define an optimization criterion. The optimization criterion used should depend on the timing of the disruption. If it occurs before the start of the semester, one may want to take advantage of this disruption to further improve the solution even if it causes more perturbations in the original solution. On the other hand, if the disruption occurs during the semester, one wants to minimize the possible impact. In this work, we consider three different optimization criteria (see Example 3):

– *Hamming distance (HD)*: The **goal** is to minimize the number of lectures that have different assignments (both room and time assignments). Domain



	Mon	Tue	Wed	Thu	Fri
8:00					
8:30			B^{PB}		
9:00			(R2)		
9:30	A^{T}			A^{T}	B^{T}
10:00	(R1)		B^{T}	(R1)	(R1)
10:30			(R1)		C^{T}
11:00	A^{L}		D^{T}	D^{PB}	(R1)
11:30	(L1)	A^{L}	(R1)	(R2)	D^{T}
12:00	C^{T}	(L1)	C^{T}		(R1)
12:30	(R1)		(R1)	D^{PB}	
13:00	D^{T}	C^{PB}		(R2)	B^{PB}
13:30	(R1)	(R2)			(R2)

(a) Optimization Criteria: HD and COM.



	Mon	Tue	Wed	Thu	Fri
8:00					
8:30			B^{PB}		
9:00			(R2)		
9:30	A^{T}			A^{T}	B^{T}
10:00	(R1)		B^{T}	(R1)	(R1)
10:30			(R1)		C^{T}
11:00	A^{L} D^{PB}		D^{T}	D^{PB} A^{L}	(R1)
11:30	(L1) (R2)		(R1)	(R2) (L1)	D^{T}
12:00	C^{T}		C^{T}		(R1)
12:30	(R1)		(R1)		
13:00	D^{T}	C^{PB}			B^{PB}
13:30	(R1)	(R2)			(R2)

(b) Optimization Criterion: COM.

Fig. 2: Two different timetables for a class of students after occurring two disruptions: (i) an unavailability constraint over the room L2; (ii) a no overlap constraint relating to the two lab lecture of A. The colors represent the different rooms were the lectures are assigned.

independent metrics, such as this one, are not fair or equitable for all actors involved (Phillips et al. 2017).

- *Weighted Hamming distance (WHD)*: WHD is a weighted version of the Hamming distance. This metric allows us to compute the number of students affected by the perturbations. Therefore, the **goal** is to minimize the impact of the perturbations in the student's academic performance. This criterion is applied at IST to evaluate the new timetable when disruptions occur during the semester.
- *Compact timetable (COM)*: The **goal** is to minimize the number of gaps in a student's timetable. This criterion is applied at IST to evaluate the new timetable at the beginning of the semester. In the past, a similar metric has been proposed (Vermuyten et al. 2016; Burke et al. 2010). However, our metric refers to the student's timetable and not a pre-defined curriculum.

Example 3 Let us consider again the Example 1 shown in Figure 1a, with the disruptions: (i) the classroom L2 is closed and (ii) the lecture A^{L} cannot be overlapped. These disruptions cause the solution to be infeasible. As explained above the optimal solution to MPP has the value two, if one applies the Hamming distance. However, this may not be the best approach. When generating the new timetable at the beginning of the semester one can use the disruption to improve the overall quality. At IST, the academic services try to use the disruptions to improve the value of COM.

There is more than one optimal solution to this problem when considering the Hamming distance. The solution shown in Figure 1b is one. Another possible solution is to change A^{L} from Tuesday 10:30 to Tuesday 11. This allows the schedule for a student to be improved, as it reduces the number of gaps. This solution is shown in Figure 2a.

However, the timetable has yet room for improvement. If one assumes that the group of students that attend D^{PB} and A^{L} is disjoint, one can overlap these lectures, which would allow the course of A^{L} to be always preceded by A^{T}. This would also reduce the time students must spend at university. On the other hand, this solution reduces the number of choices a student can make. This solution is shown in Figure 2b and requires more perturbations.

4 Modeling

In this section, we present two different models to solve the university timetabling problem described above. Table 3 summarizes the constraints used in the different models and the encoding for the most common disruptions. An in-depth description of the models is presented below.

Let us consider an arbitrary decision variable y, where y and $\overline{y}$ represent the same decision variable in different conditions. y and $\overline{y}$ are decision variables for the original problem and MPP respectively. The HD optimization criterion counts the number of decision variables that changed value $y \neq \overline{y}$, independently

Table 2: Minimization statements for the Boolean and mixed models.

	BOOLEAN	MIXED
HD	$\sum_{t\in T,d\in D,l\in L} a^l_{t,d} \neq \overline{a^l_{t,d}} +$ $\sum_{r\in R,l\in L} x_{l,r} \neq \overline{x_{l,r}}$	$\sum_{l\in L} a^l \neq \overline{a^l}$ + $\sum_{r\in R,l\in L} x_{l,r} \neq \overline{x_{l,r}}$
WHD	$\sum_{t\in T,d\in D,l\in L} std_l \times (a^l_{t,d} \neq \overline{a^l_{t,d}}) +$ $\sum_{r\in R,l\in L} std_l \times (x_{l,r} \neq \overline{x_{l,r}})$	$\sum_{l\in L} std_l \times (a^l \neq \overline{a^l})$ + $\sum_{r\in R,l\in L} std_l \times (x_{l,r} \neq \overline{x_{l,r}})$
COM	$\sum_{s\in S} \sum_{d\in D} \sum_{t\in T} c_{s,d,t}$	$\sum_{s\in S} \sum_{l\in S_l} c_{s,l}$

of their meaning and domain. Auxiliary variables are not considered in the optimization since it would add biases to the result. The WHD optimization criterion adds a weight (the number of students affected by the change in a decision variable) to the previous criterion. Finally, COM counts the number of gaps in a student's timetable. This value is computed through the usage of auxiliary variables (described below). Table 2 summarizes the optimization statements used in the different models.

4.1 Boolean Model

In the past, different integer-programming models have been proposed to solve university timetabling problems (Burke et al. 2008; Lindahl et al. 2019; Lemos et al. 2019). These models typically use Boolean variables to indicate the assignment of lectures to rooms and time slots.

The model proposed by Lemos et al. (2019) did not allow to change the lecture's schedule. Therefore, this model had only one Boolean decision variable representing the assignment of a lecture to a room.

In the other hand, the models proposed by Burke et al. (2008) and applied by Lindahl et al. (2019) use a three-index Boolean variable to indicate the assignment of lectures to rooms and time slots at the same time. However, the models are considerably simpler. For example, they consider all lectures to have the same unitary duration. The usage of a three-index model, for our case study, is inefficient since it requires a large number of variables ($|P| \times |L| \times |R|$) of which most of them are always with the value 0.

Nevertheless, these models can be easily generalized to solve the problem at hand. However, this model requires a quadratic constraints. One can try to mitigate the cost of the quadratic constraints by linearizing these constraints with the use of auxiliary variables and constraints **?**. To this model, we called Boolean'.

4.1.1 Decision Variables

The schedule of a lecture (day and time slots) is represented with an incidence matrix a, where $a^l_{d,t}$ equals 1 if and only if a lecture l is scheduled in the time slot $t \in T_l$ of day $d \in D_l$. The assignment of a lecture to a room is represented by the Boolean variable $x_{l,r} \in \{0, 1\}$; it is equal to 1 if and only if the lecture $l \in L$ is assigned to room $r \in R$.

4.1.2 Auxiliary Variables

The variable $c_{s,d,t}$ is equal to 1 if and only if the timetable of student s has a transition from occupied (attending a lecture) to free or vice-versa.

4.1.3 Constraints

Table 3 summarizes the constraints used, that can be explained as follows. Constraint 1 ensures that each lecture $l \in L$ is assigned to len_l time slots in $|D_l|$ days. Therefore, the variable $a^l_{d,t} \forall_{d\in D_l, t\in T_l}$ is assigned the value 1 exactly $|D_l| \times len_l$ times.

Constraint 2 ensures that the assignment of time slots for lecture must be consecutive without gaps. For example, if a lecture is taught at $t = 1$ and in time slot $t_2 = 5$ all slots in between must also be occupied $(2, 3, 4)$ by the same lecture. Note that, the previous example is only feasible if and only if $len_l = 5$.

Constraint 3 ensures that each lecture $l \in L$ with $R_l \neq \emptyset$ must be assigned to exactly one room. Constraint 4 ensures that at-most one lecture from a student's timetable in taught in each time slot of a day. Constraint 5 ensures that no room can have at-most one lecture scheduled per time slot per day. Constraint 6 ensures that at-most $overlap_{L^i_c}$ lectures from the same course can be lectured at the same time. Constraint 7 ensures that, in the worst case, only $\alpha\%$ of the students enrolled may not be seated.

Finally, to encode disruptions in this model one needs the following constraints. Constraint 8 ensures that it is impossible to assign lecture l to the room r. Constraint 9 ensures that lecture l cannot be assigned in the slot t of the day d. Constraint 10 ensures that two

Table 3: Constraints in the Boolean and mixed models.

	BOOLEAN	MIXED
1.	$\forall_{l\in L} \sum_{d\in ,D_l} \sum_{t\in ,T_l} a^l_{d,t} = len_l$	$\forall_{l\in L}\ a^l \geq 0$
2.	$\forall_{l\in L,d\in D,t,t_1,t_2\in T,t<t_1<t_2}$ $a^l_{d,t_1} = 1$ iff $a^l_{d,t} + a^l_{d,t_2} = 2$	$\forall_{l\in L}$ $a^l + len_l \leq \text{floor}(\frac{a^l}{\lvert T\rvert} + 1) \times \lvert T\rvert$
3.	$\forall_{l\in L} \sum_{r\in R_l} x_{l,r} = 1$	
4.	$\forall_{s\in S,d\in D,t\in T}$ $\sum_{l\in L_s} a^l_{d,t} \leq 1$	$\forall_{l_1\in L_s,l_2\in L_s,s\in S}$ $v_{l_1,l_2} + v_{l_2,l_1} \geq 1$
5.	$\forall_{r\in R,d\in D,t\in T}$ $\sum_{l\in L} x_{l,r} \times a^l_{d,t} \leq 1$	$\forall_{l_1\in L,l_2\in L}$ $s_{l_1,l_2} \times (v_{l_1,l_2} + v_{l_2,l_1}) \leq s_{l_1,l_2}$
6.	$\forall_{d\in D,t\in T,c\in C}$ $\sum_{L^i_c\in L_c} \sum_{l\in L^i_c} a^l_{d,t} \leq overlap_{L^i_c}$	$\forall_{c\in C}$ $\sum_{L^i_c\in L_c} \sum_{l_1,l_2\in L^i_c} o_{l_1,l_2} \leq overlap_{L^i_c}$
7.	$\forall_{l\in L,r\in R}\ (std_l - std_l \times \alpha) \times x_{l,r} \leq cap_r$	
8.	$x_{l,r} = 0$	
9.	$a^l_{d,t} = 0$	$a^l\ != (d \times \lvert T\rvert) + t$
10.	$\forall_{d\in D,t\in T}\ a^{l_1}_{d,t} = a^{l_2}_{d,t}$	$a^{l_1} = a^{l_2}$
11.	$\forall_{l\in L}\ x_{l,r} = 0$	

lectures are assigned at the same time slots in the same days. Constraint 11 ensures that no lecture is assigned to room r.

4.2 Mixed Model

When creating mixed integer programming models for job-shop scheduling (Ku and Beck 2016) it is common to define an integer variable to define the start time of a job on a machine. One can propose an integer model with only one integer variable to define the start period ($|P|$) of a lecture on a room. An integer model would require fewer variables than the Boolean model. However, it would also cause unnecessary constraints since some constraints do not depend on both lecture-room and lecture-time assignments. Moreover, it makes the application of decomposition techniques impossible (Lach and Lübbecke 2008).

For this reason, we propose the mixed model that uses a Boolean variable for the assignment of rooms and an integer variable for the schedule.

The global number of variables of the mixed model is smaller when compared to the Boolean model since we only need an integer variable for each lecture instead of a Boolean variable for each time slot. This model removes some constraints from the Boolean model. The implementation ensures that the constraint of *room conflicts* is not quadratic through the use of auxiliary variables.

4.2.1 Decision Variables

The starting period of a lecture l is represented by an integer variable $a^l \in [0, \dots, |P|]$. The assignment of a lecture to a room is described by the Boolean variable $x_{l,r} \in \{0, 1\}$. It is equal to 1 if and only if lecture $l \in L$ is assigned to room $r \in R$.

4.2.2 Auxiliary Variables

Variable v_{l_1,l_2} is equal to 1 if and only if l_1 is taught before l_2. Variable o_{l_1,l_2} is equal to 1 if and only if l_1 is taught at the same time as l_2. Variable s_{l_1,l_2} is equal to 1 if and only if l_1 is taught in the same room than l_2. Finally, variable $c_{l,s} \in \{0, 1, 2\}$ represents the number of transitions from free to occupied in the timetable of student s before/after lecture l.

4.2.3 Constraints

Table 3 summarizes the constraints used, that can be explained as follows. First, one needs to ensure that all lectures have a valid start period (Constraint 1).

In the Mixed model, the usage of an integer variable for the schedule ensures the lecture is taught in consecutive time slots. However, it is necessary to ensure all slots of the same lecture are taught on the same day (Constraint 2). Therefore, the last time slot of the lecture has to be taught until the last time slot of the day. The first free time slot after the end of the lecture l

is $a^l + len_l$. The expression floor$(\frac{a^l}{|T|} + 1)$ determines the day of the week from the period. Now, if one multiplies the day with the number of time slots per day ($|T|$) one obtains the first time slot of the next day.

Constraint 3 ensures that each lecture has exactly one room assigned (as explained above).

Constraint 4 ensures that all students can attend the lectures for which they are enrolled. This constraint uses the auxiliary variable v_{l_1,l_2} to ensure that l_1 is taught before or after l_2. Therefore, lectures l_1 and l_2 can never overlap.

Constraint 5 deals with *Room conflicts*. Once again we use the auxiliary variable v_{l_1,l_2} to ensure order between the lectures. In this case, the constraint is only applied when the lectures are assigned to the same room (s_{l_1,l_2}).

For Constraint 6, we use the auxiliary variable o_{l_1,l_2} which ensures that two lectures are taught at the same time. o_{l_1,l_2} can be defined by ensuring the result of adding v_{l_1,l_2} with v_{l_2,l_1} is zero.

Constraint 7 ensures that the room's capacity is respected (as explained above for the BOOLEAN model).

Finally, to encode disruptions in this model one needs the following constraints. Constraint 8 ensures that it is impossible to assign lecture l to the room r. For Constraint 9 one needs to convert the time slot $t \in T$ to periods by summing t with the result of multiplying the day d with the number of time slots per day. This way one can ensure that lecture l cannot be assigned in the slot t of the day d. The two last constraints can be easily modified to encode *preference time* and *room preference* disruptions. Constraint 10 ensures that two lectures have the same start period. Constraint 11 ensures that no lecture is assigned to room r.

5 Evaluation

In this section, we examine the quality of the proposed models using the real-life instances described below. To characterize these data sets, it is important to define room *frequency* (Beyrouthy et al. 2009) and room *utilization* (Beyrouthy et al. 2009). Room *frequency* denotes the ratio between the number of time slots used and the total number of time slots available. Room *utilization* denotes the ratio between the number of seats used and the total number of seats available considering all time slots in all days.

Table 4 shows a comparison between the data sets present in the literature. One can observe that the data set from IST is considerably larger than the data sets from ITC and KUL. Furthermore, the data sets from ITC and KUL consider all lectures with the same length which can significantly simplify the encoding. When analyzing the data set from IST, one can see that room *utilization* is higher than room *frequency*. This can be easily explained by the need for lectures with overbooking (Lemos et al. 2019) (*i.e.* more students seated than actual seats).

5.1 Experimental Setup

The solution was tested using the data sets from the courses taught in the Taguspark campus of IST. Note that, the data set from KUL does not specify different capacities for each room. For this reason, the *utilization* metric cannot be applied. Note that, the **Taguspark** instances are going to be referenced from now on only by FALL and SPRING. We also have the current hand-made solution that can be used as starting point for the search algorithm. The values for α were previously obtained in Lemos et al. (2019) by optimizing this parameter. The data is encoded with ITC-2019 XML format (Müller et al. 2018). The code and data sets are available at `https://github.com/ADDALemos/MPPTimetables/releases/tag/J.Scheduling19`. The data used to test the system was obtained through the FenixEdu^{TM}system public API[3] which is in use at IST.

The tests were run using the *runsolver* tool (Roussel 2011) with a time out of 600 seconds and a limit of 3 GB of memory. The tests were performed on a computer running Ubuntu 14, with 24 CPUs at 2.6 GHz and 64 Gb of RAM. The program was implemented in C++, using the XML parser *RAPIDXML*[4] to read the timetabling. The models were implemented using *Gurobi* (Gurobi Optimization 2018). The *Gurobi* solver was run with parameters *GRB_IntParam_Threads = 3* and *GRB_IntParam_Presolve = 0* for best performance.

5.2 Generating Disruptions

To test our approach, we first analyzed the timetables from the last 5 years to identify which disruptions are the most common. These disruptions generally occur at the beginning of the new semester when a new timetable is generated. To compute the likelihood of a disruption to occur, we take into account the number of perturbations (Hamming distance) occurring from one year to another, over the total number of variables.

The average percentage of disruptions by type is shown in Table 5. Note that *time* and *room assignment* represent the probability of a specific lecture to have a

[3] http://fenixedu.org/dev/api/

[4] *RAPIDXML* is available from `http://rapidxml.sourceforge.net/manual.html`

Table 4: Data sets characteristics.

	ITC-2007 (Di Gaspero et al. 2007)	KUL (Vermuyten et al. 2016)	Taguspark (Lemos et al. 2019)	
			SPRING	FALL
# Days	5	5	5	5
# Slots per day	5.38	6	26	26
# Students[a]	5606.2	19690	31833	31144
# Courses	89	UKN[b]	147	163
# Lectures per Week	313.77	396	751	728
AVG Slots per Lecture	1	1	2.9	2.99
STDEV Slots per Lecture	0	0	0.5	0.52
AVG Enrollment per Lecture	61.67	49.7	77.45	77.22
STDEV Enrollment per Lecture	45.78	37	79.7	79.94
# Rooms	16.08	56	43	43
AVG Capacity by Room	114.71	UKN[c]	52.74	52.74
STDEV Capacity by Room	91.35	UKN	51.56	51.56
Frequency	76.42	13.2	49.36	39.56
Utilization	41.61	-	64.74	54.63

[a] This value may include double counting of the students.
[b] KUL generates the timetables post-enrollment. Therefore, the information about the courses is not necessary.
[c] It is considered that the rooms have sufficient capacity for the lectures.

Table 5: Average percentage of disruptions in the last 5 years.

Type of Disruption	Preference/Invalid Time Assignment	Preference/Invalid Room Assignment	Modify Enrolments Up (Down)	Overlap	Modify Number of Lectures Insert (Remove)	Insert Curriculum
%	21	25	25 (27)	11	14 (8)	<1

preference or a constraint in time or room assignment, respectively. We use these results to randomly generate the number of disruptions of the same type. The disruptions were randomly generated following a uniform distribution.

The information is shown in Table 5 is not enough for generating the full set of disruptions. In some cases, it is also necessary to correctly quantify the perturbations in a given lecture/course. This is the case of two specific disruptions: *modify enrollments* and *modify number of lectures.*

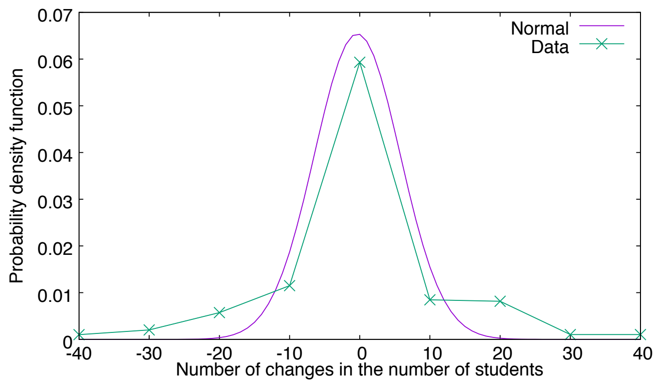
For example, when modifying enrollments we need to decide in which lectures the number of students changes (based on the value from Table 5) and the actual number to increase/decrease. One could analyze the changes and uniformly select one of these values since these values are not equiprobable. Therefore, we used the Microsoft Excel Solver (Fylstra et al. 1998) to estimate the parameters of a normal distribution that would best fit our data sets. Figure 3a shows the comparison between the closest-fit normal distribution and our data set.

The same process can be applied to find the correct distribution for the disruption *modify number of lectures.* Figure 3b shows the comparison between the closest-fit normal distribution and our data set.

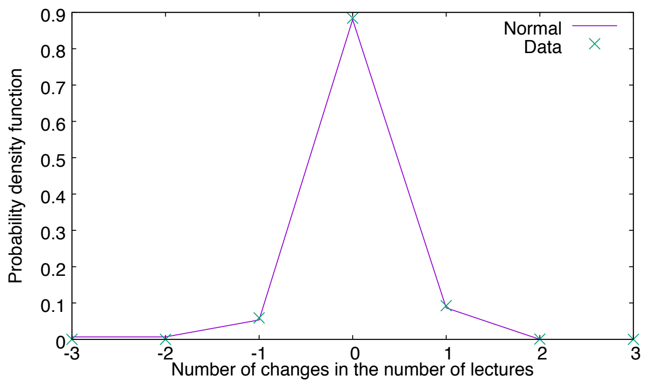
5.3 Computational results

5.3.1 Course Timetabling problem

In a first approach, we compare the performance of the two different models (proposed in Section 4) when solving the original version of the problem, *i.e.* before disruptions occur. Naturally, we considered the COM as the optimization criteria. To improve performance, we apply a warm-start procedure using the hand-made solution. Note that, despite the fact the academic services use COM as the optimization criteria, the hand-made solution is feasible but not optimal. We also tested using greedy heuristics (the extension from our work (Lemos



(a)



(b)

Fig. 3: Normal distribution that best fits the data sets of fluctuations in the (a) students enrollments (with mean of -0.3 and standard deviation of 6), and (b) the number of lectures (with mean of 0.04 and standard deviation of 0.45).

Table 6: Results for BOOLEAN, BOOLEAN' and MIXED models solving the course timetabling problem with warm-start, considering general (Gen.) and quadratic (Qua.) constraints.

	FALL			SPRING		
	BOOLEAN	BOOLEAN'	MIXED	BOOLEAN	BOOLEAN'	MIXED
CPU Time (s)	Time Out	Time Out	32.17	Time Out	Time Out	22.69
Optimal	N/A	N/A	Found	N/A	N/A	Found
# Variables	5269200	5280380	8487526	5255628	5266548	1668712
# Gen. Constraints	5200000	5213975	2111238	5187000	5187000	1810312
# Qua. Constraints	5590	0	0	5460	0	0

et al. 2019)). However, they are worse in terms of quality and similar in terms of CPU time. Therefore, we do not show the result.

Table 6 shows the results for the different models (BOOLEAN, BOOLEAN' and MIXED) when solving the course timetabling problem with warm-start. Observe that the BOOLEAN model requires both general and quadratic constraints. The linearized version of the BOOLEAN model only requires general constraints. However, it adds two new auxiliary variables for each quadratic constraints. The BOOLEAN' model also adds two and half more constraints for each new variable. There is no clear advantage in the BOOLEAN model versus BOOLEAN' model with this time limit. The MIXED model requires fewer constraints and variables. In terms of CPU time, the BOOLEAN model is not able to find an optimal solution within the 600 seconds limit.

It is possible to reduce the size of the problem by reducing the number of students since not all students affect the solution. In this case, the students are used to identify the curricular path. It is natural that groups of students attend the same lectures. Therefore, one does not need to generate constraints for all students. The constraints can be generated only for *distinct* students. In the data sets, around 15% of all students attend exactly the same lectures. This number is relatively small but allows us to reduce the total number of constraints by 10% on average for each instance.

To study the impact of the warm-start strategy in the performance, we run the tests with and without this strategy for the MIXED model. The results from FALL and SPRING improve from 1822.57 and 1726.21 to 32.17 and 22.69, respectively. The version with warm-start is two orders of magnitude faster. The hand-made solution, despite not being optimal, is shown to be a good starting point.

5.3.2 Minimal Perturbation Problem

The models proposed in this paper were also tested in the presence of disruptions. For each disruption type, 50 different instances were created. In this paper, we only show the results for the most relevant disruptions (*i.e.* the disruptions that are likely to occur). The instances were generated based on the data analyzes explained

Table 7: Results for the most common disruptions using the MIXED model. TO stands for time out, δ_{HD} measures the number of perturbations, δ_{WHD} measures the number of students affected by the perturbations and δ_{COM} measure the change in the number of gaps in the student's timetable.

		Modify Enrollments		*Invalid Time*		*Invalid Room*	
Opt.		FALL	SPRING	FALL	SPRING	FALL	SPRING
COM	AVG. Time (s)	46.2	20.65	TO	TO	73.36	60.52
	Median Time (s)	46.23	20.16	TO	TO	72.34	59.95
	AVG. δ_{COM}	-1	-1.5	N/A		0.5	0
	COM % Optimal	100	100	N/A		80	90
	AVG. δ_{HD}	510	399	N/A		372	340
	AVG. δ_{WHD}	27303.9	27287.1	N/A		27000	27602.5
	% of Feasible Solution	100	100	N/A		80	90
HD	AVG. Time (s)	40.79	42.91	TO	TO	74.45	54.37
	Median Time (s)	44	42.99	TO	TO	75	51.9
	AVG. δ_{COM}	0	0	N/A		4.4	0.87
	AVG. δ_{HD}	489	380	N/A		300	309.75
	HD % Optimal	100	100	N/A		80	90
	AVG. δ_{WHD}	26803.5	26990.2	N/A		26660.83	27334.7
	% of Feasible Solution	100	100	N/A		80	90
WHD	AVG. Time (s)	39.97	43.6	TO	TO	69.17	60.28
	Median Time (s)	44	43.99	TO	TO	68.15	62.95
	AVG. δ_{COM}	0	0	N/A		4.2	0.6
	AVG. δ_{HD}	508	399	N/A		490	489
	AVG. δ_{WHD}	25908.5	26512.9	N/A		26442	27468.12
	WHD % Optimal	100	100	N/A		80	90
	% of Feasible Solution	100	100	N/A		80	90

above. We tested the models using all the disruptions found on the original data (see Table 5). In this section, we discuss the most relevant disruptions.

Once again and as expected, the MIXED model outperforms the BOOLEAN model. For this reason, only the results for the MIXED model are shown in Table 7. In order to improve performance, the warm-start strategy was modified to only add a partial solution (the variables affected by the disruption are not added).

The SPRING instance is easier to solve since it has a smaller number of lectures to assign. This fact can be explained by the organization of the curricular plans.

When analyzing the performance one can see that the *modify enrollment* disruption instances are the easiest to be solved. This can be explained by the fact the disruption adds, in general, only a small number of students. The use of slack in room capacities also contributes to better performance. This is the only disruption that is able to improve the value of the COM (when optimizing this criterion). This disruption also allows reducing the number of students enrolled. *Modify enrollments* is the only disruption to have all instances with a feasible solution. The HD and WHD optimization criteria reduce the number of perturbations and the number of students affected while not improving the COM. For this disruption, the value of COM stays equal to the one found in the original solution.

The disruptions *invalid time assignment* and *invalid room assignment* cannot lead to a solution with an improved optimization value, given that these disruptions only add new constraints to the problem. The results show that the quality of the solution gets worse with these disruptions. Note that these disruptions may cause the instance to be infeasible. For all instances with feasible solutions, our approach finds optimal solutions. The *invalid time assignment* disruption is the only one to reach the time limit before finding a solution.

Globally, one can see that optimizing using COM causes more perturbations in the solution and affects more students. This result confirms the results obtained by Lindahl et al. (2019), where it is said that performing more perturbations can improve the quality of the solution. Optimizing based the WHD criteria also causes more perturbations but reduces the number of students affected by the change.

5.3.3 Incremental Approach for Recovery after Disruption

The simple recovery process takes too long for the *invalid time assignment* disruption. Therefore, we propose an incremental approach to reduce the search space by splitting the problem into two stages. A disruption of the type *invalid time assignment* clearly causes a change in the time slots of the lecture. However, it may not be necessary to modify the room assignment. For this reason, we divide the problem into: (i) the problem of assignment of lectures to rooms and (ii) the whole problem. The first stage has to deal with all the constraints relating to time slots, where one has to consider a solution to the problem of assigning rooms to lectures. In this case, the original solution for this subproblem is added to the model as static. The second stage applies a warm-start with the results of the first stage, possibly improving its quality. This division does not exclude possible solutions since the second stage includes the whole problem again. This decomposition can be changed (*i.e.* start with the problem of assigning lectures to rooms) depending on the disruption since we know beforehand which part of the problem must change.

Naturally, the proposed decomposition does not provide any guarantees about optimality at the end of the first stage. Nevertheless, this decomposition ensures that a feasible solution to the first stage is a feasible solution to the whole problem. Conversely, one cannot conclude anything about the infeasibility of the instance based on the result from the first stage.

Table 8 shows the results for the *invalid time assignment* disruption, where one can see that the incremental process is much faster. In this case, the second stage does not improve the quality of the solution. In other words, the first stage not only found a feasible solution but also found an optimal solution to the complete problem.

6 Conclusion and Future Work

This paper discusses the problem of solving university timetabling problems after the occurrence of disruptions. In particular, the methods proposed to solve the MPP for real-world university timetabling problems.

The two integer programming models proposed can find the *closest* new feasible solution for most common university timetabling disruption scenarios. Our approaches were successfully evaluated with data sets from IST. To correctly test the performance of the methods we randomly generated disruptions with the probability learned from the history of the last five years of timetables. The models were tested using different optimization criteria, including the optimization criteria applied at the academic offices at IST.

To improve the performance, we propose an incremental algorithm that divides the problem into two stages: the problem of assignment of lectures to rooms and the whole problem. The incremental algorithm can reduce the size and execution time without losing quality. Moreover, the experimental results also show that using warm-start techniques in this type of problems provide a significant advantage. We warm-start the algorithm with a partial feasible solution extracted from the solution of the previous year.

As future work, we propose to extend this method in order to exploit the incremental nature of MPP. This way, every time a new disruption arises "only" another step in the incremental integer programming algorithm is required. The usage of incremental algorithms would reduce CPU time by avoiding the repetition of redundant steps when searching for a new feasible solution.

References

Achá RJA, Nieuwenhuis R (2014) Curriculum-based course timetabling with SAT and maxsat. Annals Operations Research 218(1):71–91, DOI 10.1007/s10479-012-1081-x, URL https://doi.org/10.1007/s10479-012-1081-x

Banbara M, Inoue K, Kaufmann B, Okimoto T, Schaub T, Soh T, Tamura N, Wanko P (2019) teaspoon : solving the curriculum-based course timetabling problems with answer set programming. Annals Operations Research 275(1):3–37, DOI 10.1007/s10479-018-2757-7, URL https://doi.org/10.1007/s10479-018-2757-7

Bellio R, Gaspero LD, Schaerf A (2012) Design and statistical analysis of a hybrid local search algorithm for course timetabling. Journal Scheduling 15(1):49–61, DOI 10.1007/s10951-011-0224-2, URL https://doi.org/10.1007/s10951-011-0224-2

Bettinelli A, Cacchiani V, Roberti R, Toth P (2015) An overview of curriculum-based course timetabling. TOP 23(2):313–349, DOI 10.1007/s11750-015-0366-z, URL https://doi.org/10.1007/s11750-015-0366-z

Beyrouthy C, Burke EK, Landa-Silva D, McCollum B, McMullan P, Parkes AJ (2009) Towards improving the utilization of university teaching space. Journal of the Operational Research Society 60(1):130–143

Burke EK, Mareček J, Parkes AJ, Rudová H (2008) Penalising patterns in timetables: Novel integer programming formulations. In: Operations Research Proceedings, Springer, pp 409–414, DOI 10.1007/978-3-540-77903-2_63, URL https://doi.org/10.1007/978-3-540-77903-2_63

Table 8: Incremental approach to recover after disruptions of the type *invalid time*. δ_{HD} measures the number of perturbations, δ_{WHD} measures the number of students affected by the perturbations and δ_{COM} measure the change in the number of gaps in the student's timetable.

	FALL		SPRING	
	Stage 1	Stage 2	Stage 1	Stage 2
Avg Time (s)	239.02	68.67	210.68	58.4
Median Time (s)	282	68	231.24	58
Avg δ_{COM}	2	2	1	1
AVG δ_{HD}	416.67	416.67	400.34	400.34
AVG δ_{WHD}	26726.34	26726.34	27473.34	27473.34
% Feasible Solution	85		90	

Burke EK, Marecek J, Parkes AJ, Rudová H (2010) Decomposition, reformulation, and diving in university course timetabling. Computers & Operations Research 37(3):582–597, DOI 10.1016/j.cor.2009.02.023

Cacchiani V, Caprara A, Roberti R, Toth P (2013) A new lower bound for curriculum-based course timetabling. Computers & Operations Research 40(10):2466–2477, DOI 10.1016/j.cor.2013.02.010

Di Gaspero L, Schaerf A, McCollum B (2007) The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Tech. rep., Queen's University

Even S, Itai A, Shamir A (1976) On the complexity of timetable and multicommodity flow problems. Society for Industrial and Applied Mathematics SIAM Journal on Computing 5(4):691–703

Fylstra DH, Lasdon LS, Watson J, Waren AD (1998) Design and use of the microsoft excel solver. Interfaces 28(5):29–55, DOI 10.1287/inte.28.5.29, URL `https://doi.org/10.1287/inte.28.5.29`

Gurobi Optimization L (2018) Gurobi optimizer reference manual. URL `http://www.gurobi.com`

Hamming RW (1950) Error detecting and error correcting codes. The Bell System Technical Journal 29(2):147–160, DOI 10.1002/j.1538-7305.1950.tb00463.x

Kingston JH (2013) Educational timetabling. In: Automated Scheduling and Planning - From Theory to Practice, pp 91–108, DOI 10.1007/978-3-642-39304-4_4, URL `https://doi.org/10.1007/978-3-642-39304-4_4`

Ku W, Beck JC (2016) Mixed integer programming models for job shop scheduling: A computational analysis. Computers & Operations Research 73:165–173, DOI 10.1016/j.cor.2016.04.006, URL `https://doi.org/10.1016/j.cor.2016.04.006`

Lach G, Lübbecke ME (2008) Optimal university course timetables and the partial transversal polytope. In: Experimental Algorithms, 7th International Workshop, pp 235–248, DOI 10.1007/978-3-540-68552-4_18

Lemos A, Melo FS, Monteiro PT, Lynce I (2019) Room usage optimization in timetabling: A case study at universidade de lisboa. Operations Research Perspectives 6:100092, DOI https://doi.org/10.1016/j.orp.2018.100092, URL `http://www.sciencedirect.com/science/article/pii/S2214716018301696`

Lindahl M, Stidsen T, Sørensen M (2019) Quality recovering of university timetables. European Journal of Operational Research DOI https://doi.org/10.1016/j.ejor.2019.01.026, URL `http://www.sciencedirect.com/science/article/pii/S0377221719300451`

McCollum B (2006) University timetabling: Bridging the gap between research and practice. In: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Springer, pp 15–35

Müller T (2009) ITC-2007 solver description: a hybrid approach. Annals of Operations Research 172(1):429

Müller T, Rudová H, Barták R (2004) Minimal perturbation problem in course timetabling. In: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT), pp 126–146, DOI 10.1007/11593577_8

Müller T, Rudová H, Müllerová Z (2018) University course timetabling and international timetabling competition 2019. In: Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT), p 27

Phillips AE, Walker CG, Ehrgott M, Ryan DM (2017) Integer programming for minimal perturbation problems in university course timetabling. Annals Operations Research 252(2):283–304, DOI 10.1007/s10479-015-2094-z, URL `https://doi.org/10.1007/s10479-015-2094-z`

Pillay N, Özcan E (2019) Automated generation of constructive ordering heuristics for educational timetabling. Annals Operations Research 275(1):181–208, DOI 10.1007/s10479-017-2625-x, URL `https://doi.org/10.1007/s10479-017-2625-x`

Roussel O (2011) Controlling a solver execution with the runsolver tool. Journal on Satisfiability, Boolean Modelling and Computation 7(4):139–144

Sakkout HE, Wallace M (2000) Probe backtrack search for minimal perturbation in dynamic scheduling. Constraints 5(4):359–388, DOI 10.1023/A:1009856210543, URL `https://doi.org/10.1023/A:1009856210543`

de Souza Rocha W, Claudia M, Boeres S, Rangel MC (2012) A grasp algorithm for the university timetabling problem. In: Proceeding of 9th International Conference of the Practice and Theory of Automated Timetabling (PATAT), pp 404–406

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, DOI 10.1016/j.ejor.2015.11.028

Vrielink RAO, Jansen EA, Hans EW, van Hillegersberg J (2019) Practices in timetabling in higher education institutions: a systematic review. Annals Operations Research 275(1):145–160, DOI 10.1007/s10479-017-2688-8, URL https://doi.org/10.1007/s10479-017-2688-8

Zivan R, Grubshtein A, Meisels A (2011) Hybrid search for minimal perturbation in dynamic csps. Constraints 16(3):228–249, DOI 10.1007/s10601-011-9108-5, URL https://doi.org/10.1007/s10601-011-9108-5