

Louvain School of Management

The University Course Timetabling Problem: A case study

Design of the LSM course timetable using an
Iterated Local Search

Auteur : Edouard Dive
Promoteur : Daniele Catanzaro
Année académique 2019-2020

I Abstract

Every year, universities face the challenge to build a course timetable which due to their large size, usually require a huge amount of time and human resources. In this thesis, we develop a model of an automated course timetable for the Louvain School of Management (LSM-UC Louvain). To do so, a hybridised algorithm has been designed and implemented in Mosel language. The model is composed of two phases. The first stage consists in generating a feasible initial solution using the combination of a Graph Colouring algorithm and a Maximum Matching algorithm. The second phase consists of applying a metaheuristic called "Iterated Local Search" to improve the initial solution until reaching a satisfactory solution timetable ready to use. That strategy has produced effective and promising results, and the best solution obtained seemed to outperform the real LSM timetable value currently used. However, further improvements and data refinement may be necessary to be used by the LSM.

Key words: Operational Research, Metaheuristic, Course timetabling, University, Optimisation, Iterated Local Search

Google Drive link: [Click here](#).

II Acknowledgments

I want to first thank my supervisor, Professor Daniele Catanzaro, for the time and guidance he gave to me throughout this process. I want to emphasise his amazing way of teaching, which not only makes him probably one of the best teachers I have had at university, but which also provided me with the will to work with him.

I also want to warmly thank my American and Canadian friends, Adriana Lovera and Andrew King, who dedicated their time to proofread and to perfect the form of this thesis.

I would also like to thank my family for their unwavering support since the beginning, as well as the countless others who encourage me in so many ways .

Finally, I want to thank the UCLouvain, the LSM and my professors for the knowledge I acquired and the skills I developed that allowed me to complete this thesis and prepared me to face future challenges.

Contents

I Abstract	i
II Acknowledgments	ii
1 Introduction	1
2 Review of approached solutions for the UCTP	2
3 Data and Assumptions	5
3.1 Courses, Majors and Rooms Data	5
3.2 Assumptions	6
4 Problem Formulation, Constraints and Cost function	8
4.1 Objectives	8
4.2 Hard and Soft Constraints	8
5 Solution Method	11
5.1 Resolution Strategy	11
5.2 The Iterated Local Search Algorithm (ILS)	11
5.3 Phase 1: Generate the Initial Solution	12
5.3.1 Step 1: Pre-processing and Data Structure	13
5.3.2 Step 2: Attribution of lessons to periods	14
5.3.3 Step 3: Allocation to available rooms	16
5.3.4 Step 4: Management of eventual remaining lessons	17
5.4 Phase 2: Optimisation with an Iterated Local Search (ILS)	17
5.4.1 Local Search	17
5.4.2 Perturbation	19
5.4.3 Acceptance and Stopping Criteria	20
6 Numerical Results and Analyses	21
6.1 Results of the ILS	21
6.2 Comparison with the real LSM-2019 timetable	24
6.3 Variation of the Perturbation Strength	25
7 Critical Analysis and limitations	27
8 Conclusions	29

9	References	33
10	Annexes	34
10.1	Costs details	34
10.2	An example of best timetable found	39

1 Introduction

The challenge of building timetables or schedules is met by many institutions and companies that have to manage limited resources. That task concerns several fields like transportation (aviation, trains), manufacturing (shifts), education (courses and exams), healthcare (surgeons' and nurses' timetables) or sports (planning of matches) (Petrovic & Burke, 2004). For instance, airports have to continuously attribute a landing track and a gate to each plane waiting in the line, without compromising security and planned schedules. A production factory is challenged daily to optimise the productivity of its machinery resources, improving the well-being of its employees and reducing wasted time and money with adapted schedules. The supply chain sector experiences scheduling and resource allocation problems at every stage of the logistics. Understandably, institutions like schools and universities regularly face the enormous task of assigning lectures to classrooms and timeslots, while avoiding courses conflicts, respecting teachers' schedules, to name but a few of the challenges.

This Master's thesis tackles the scheduling problem faced by a university, also designated as the University Course Timetabling Problem (UCTP). Usually, universities build the course timetable manually, which takes a large amount of time and human resources every year (Alvarez-Valdes & al., 2002). However, when a feasible timetable has been found, it may be not the most optimal. Nowadays, the progress made in the numerical sector, and more precisely in the Operational Research and Artificial Intelligence fields, has provided state-of-the-art techniques to propose optimised solutions to scheduling problems. The aim of this paper is to design and implement an automated model that produces a feasible course timetable for the Louvain School of Management (LSM), based on that literature. We want to prove that a different method than the one currently used by the administration of the university can provide better results and maybe set them free from the time- and energy-consuming task to design a timetable.

This thesis is divided into seven chapters. First, a review of the techniques developed to solve the UCTP can be found in Chapter 2. Chapter 3 presents the LSM timetable data together with the assumptions done. Afterwards, a formulation of the problem is proposed in Chapter 4 along with the constraints and the tools used to evaluate the quality of a timetable. The solution method is detailed in Chapter 5 while numerical results and analyses are disclosed in Chapter 6. Finally, a critical analysis of the thesis is presented and the limitations of the model revealed.

2 Review of approached solutions for the UCTP

Building a university timetable consists of assigning courses to timeslots and classrooms. That assignment problem must respect a set of constraints that may be different depending the university. Constraints are usually divided into two categories: the hard and the soft constraints. Hard constraints cannot be violated (course conflict, room conflict) while soft-constraints must be respected as much as possible (Alvarez-Valdes & al., 2002; Petrovic & Burke, 2004). Constraints are described in detail in Chapter 3.

The University Course Timetabling Problem (UCTP) is a hybrid optimisation problem classified as a NP-hard problem (Babaei & al., 2015). That means that the problem needs an exponential number of operations to be solved, and for which no efficient resolution algorithm exists for all instances (Marmion, 2011). For such problems, exact methods can only be achievable for small problems (like maximum 10 courses) in order to get results in reasonable CPU time which is usually far from the reality that universities experience. It appears that only the use of heuristics can provide satisfactory approached solutions, with no guarantees to reach the optimal solution of the problem (Schaerf, 1999). In their survey, Babaei & al. (2015) divide heuristics providing approached solution to UCTP into 3 categories: Operational Research methods, metaheuristics methods and intelligent novel methods.

Operational Research (OR) methods includes techniques based on Graph Colouring theory, Integer Programming methods (IP) and Constraints Satisfaction Programming methods (CSP) (Babaei & al., 2015). The first researches about the university timetable were launched by Gotlieb (1963) who provided a first definition of the UCTP as three set of lecturers, rooms and timeslots. Welsh & Powell (1967) were the first to propose a solution based on the widely studied Graph Colouring Theory. That idea to assign different colours to vertices has been fostered by de Werra (1985), Selim (1988) or Razak & al. (2010). The Colour algorithm offers usually quick and feasible solutions, but the method seems to produce better results when hybridised with metaheuristics (Babaei & al., 2015). IP methods are methods whose aim is an efficient assigning of limited resources, maximising the preferences and minimising a cost function (Babaei & al., 2015). This approach has been studied by Dimopoulou & Miliotis, (2001), Daskalaki & al. (2004) and more recently Bakir & Aksop (2008). Constraints Satisfaction Programming based methods models problems as a set of variables with a finite domain. In general, those OR methods can generate feasible solutions efficiently, but the quality of the solution remains

lower than other more sophisticated methods (Abdullah, 2006). They were hybridised with meta-heuristics methods for better results, for example with a Tabu search (White & Zhang, 1997). Focacci & al. (2003) detailed a discussion about the hybridisation of the Constraints Satisfaction Programming methods.

Since the 1980s and '90s, meta-heuristic approaches have been developed for the university timetable. The advantage of those methods is that they produce very high-quality solutions but require significant tuning of parameters and have often an important computational cost (Petrovic & Burke, 2004). Meta-heuristics are split into two types: the population-based methods and the single solution-based methods.

Population-based methods start with several initial solutions, the population. A selection mechanism is used to find the best solution and depending the method used, some changes are applied over the selected solutions so that an improvement is obtained. The final performance strongly depends on the way the population is manipulated (Babaei & al., 2015; Blum & Roli, 2003). A first example applied on UCTP is the Genetic algorithm with the application of a mutation and a *cross-over* operator. Some results have been reached and can be consulted in Alsmadi & al. (2011); Blum & al. (2002); Rich (1995); Lewis & Paechter (2005). Another example is the Ant Colony algorithm (ACO) where the course timetabling is transformed in an Optimal Path problem and uses the concept of *pheromone*. Socha & al. (2002) and Dorigo & Stützle (2003, 2006) used with this method to solve the UCTP. In the comparison of metaheuristics proposed by Rossi-Doria & al. (2003), ACO method provided slightly worse performance than Simulated Annealing and Tabu Search, but better than the Genetic algorithm. Other populations methods were developed (Memetic algorithms, Harmony Search) and more information about can be found in Babaei & al. (2015).

Single solution-based methods refer to metaheuristics working on a single candidate solution instead of using an initial population. They are also referred to as *trajectory methods* (Blum & Roli, 2003). A first solution is selected based on some criteria, and while the termination criterion is not reached, a manipulation is applied until an improvement of the candidate solution is done. Tabu Search, Simulated Annealing, Explorative Local Search (Local Search, Iterated Local Search, GRASP, Variable Neighbourhood search) are some methods studied. Tabu search is among the most cited method to solve Combinatorial Optimisation problems. This method uses the history of the search, to escape from the local minima and implement an explorative strategy. A short-term memory is

implemented as a *tabu list* (Blum & Roli, 2003). Alvarez-Valdes & al. (2001) have implemented a Tabu Search to solve the course timetable of the university of Valencia and good results have been reached. Simulated annealing is a local search inspired of heating solids in physic science (Babaei & al., 2015). The idea is to allow moves providing solutions of worse quality than the current solution in order to escape from the local minima. During the search, a *temperature* variable is decreased during the process, and the more the temperature get closer to 0, the less the probability of the model to accept worse quality solutions. A solution to the UCTP is provided by Tuga & al. (2007) and Aycan & Ayav (2009). Another class of meta-heuristic methods are the exploratory strategies. A first approach is the Variable Neighbourhood Search (VNS) used by Abdullah & al. (2005) to solve the UCTP. The main idea is to apply a *Monte Carlo* acceptance criterion for improvement of the explorations, by modifying the neighbourhood structure. (Babaei & al., 2015; Blum & Roli, 2003)). Another exploratory strategy applied to UCTP has been developed by Song & al. (2018) with the use of an Iterated Local Search (ILS) that achieved highly competitive results compared with the existing algorithms. This is the reason why this method will be used to solve the LSM timetabling problem. A detailed explanation of the approach can be found in section 5.

Finally, the intelligent novel methods are another way to tackle the timetabling problem. The first set of methods are the ones designed with hybrid approaches, namely the combinations of several algorithms enumerated previously mentioned. These combinations give the opportunity to eliminate some weaknesses of the individual methods. A hybrid algorithm proposed by Kostuch (2004) use the Graph Colouring theory combined with a simulated annealing to solve the UCTP. Abdullah & al. (2007) combined a Local Search with evolutionary approaches to provide better results. Turabieh & Abdullah (2009) combined a memetic algorithm and two Tabu Search. Other hybrids and novel methods (fuzzy) can be found in Babaei & al. (2015).

Moreover, this literature review should be completed by a more recently studied approach called hyper-heuristic, which operates at a higher level of sophistication. That method is described as "*a search method or learning mechanism for selecting or generating heuristics to solve computational search problems*" (Burke & al., 2013). A hyper-heuristic runs in its algorithm a set of simple heuristics and chooses the one with the best result. A particularly promising method providing competitive results has been explored by Soria-Alcaraz & al. (2015) to solve the UCTP with an add-delete hyper-heuristic combined with an Iterated Local Search.

3 Data and Assumptions

3.1 Courses, Majors and Rooms Data

The University timetable that is going to be optimised is the one used by the Louvain School of Management (LSM), which is part of the Université Catholique de Louvain in Belgium. The timetable is based on data and information given by the LSM's secretaries. The timetable is built according to the following patterns.

Master's courses have to be given among four successive half-semester periods of six weeks spread out all over the academic year: September to October, November to December, February to March, April to May. These periods are referred to as "trimesters" in this paper. Each trimester is separated by an exam period. The exams timetable is not studied here. Each course is usually taught in one single trimester and lessons are mainly planned from Monday to Friday, between 8h30 and 18h15. On the contrary, bachelor's programs are spread out over two semesters of four months, September to December and February to May. The main consequence is that the number of hours for each course per period is divided by two in comparison with master's programs.

The LSM courses are part of 40 different academic curricula of bachelor or master (common Cores and Majors) programs and are dispatched on two different campuses: Louvain-la-Neuve and Mons. However, during each trimester, each curriculum is given on one single campus so that students must not endure long journeys in their weekly schedules, although a group of students may follow a program during Trimester 2 in LLN, while the same group could follow another in Mons during Trimester 4. Regarding masters, Trimesters 1 and 3 are mainly attributed to the core curriculum while Trimester 2 and 4 are mainly reserved for majors that students can choose. An important element is that, as much for bachelor's as for master's, a single course can be given in several programs, meaning that overlap between programs will have to be considered in order to avoid course conflicts. For the rest of this thesis, we qualify "academic programs" generally as "MAJORS" since the definition of majors and core curriculum is conceptually the same: a sub-group of courses followed by a sub-group of LSM students and where a student cannot follow two contemporary programs at the same time.

Another datapoint to take into account is the Teacher's requirement. Indeed, it is common that a teacher has to give more than one course. For courses taught to large

groups of students, the group can be split and each student allocated to one of two smaller groups for which the teacher remains the same. Two teachers can be part of one course.

Rooms data are the next bunch of constraints to consider. 160 classrooms of maximum capacities between 15 and 600 seats are available in LLN and 17 in Mons. For each room, several other details are given: the type of furniture (fixed or mobile), the location (defined as Centre, Biéreau, Hocaille, Cyclotron for LLN), and access for those with mobility challenges, for example. It should be noted that courses with a project and groupwork need a room with mobile furniture, while classic courses can be attributed to both categories.

Finally, a last table will give us for each semester the availability of each room at each timeslot of the week. Indeed, the LSM is not the only school to use the classrooms of the university. The use of rooms by other departments must be taken into account. For that, each single timetable of each room has been explored and forbidden slots added to a timetable.

3.2 Assumptions

A first assumption is about the timeslots considered per day. It has been chosen that each day will be divided into four timeslots: 8:30-10:30, 10:45-12:45, 14:00-16:00, 16:15-18:15. This organisation of timeslots is commonly used at the university of Louvain. Since the rooms occupation strongly depends on that division of daily time, this will be taken as a data. For this reason, we consider that each course is given in one or several blocks/timeslots of two hours (LESSONS). Nevertheless, two lessons of the same course can follow each other so that a room is attributed for 4 hour to a course. In reality, a course may last three hours or one hour (usually 9:30-12:30 or 14:00-17:00): the former will be considered to need a classroom during two successive timeslots, and the latter only one timeslot.

The second assumption is about the courses' variability among trimesters. We considered that each course occurs regularly during the 6 weeks constituting the Trimester. Regarding the teachers, we will consider that the only constraint is that the courses they teach do not enter into conflict given the fact that no other data were available. In addition, the room availability has also been considered as constant within each semester,

meaning that a room available during the first week will be until the sixth one.

Finally, we consider the timetable to be an array which take the "PERIODS" in rows and the "ROOMS" in columns. We define a Period as a pair composed by a day and a timeslot as used in Soria-Alcaraz's works (2015). Thus, over five days, there are twenty periods. The Solution Timetable is therefore an array of 20 rows x 160 columns for LLN and an array of 20 rows x 17 columns for MONS.

For each trimester of each campus, an Excel file has been built with four different sheets. The first one contains the COURSES data: course name and code, number of students, names of teacher(s), the rooms furniture requirements, the semester during which the course occurs and the number of lessons required per course. Figure 1 hereafter summarises the main figures of courses and majors for each trimester. The second sheet contains the MAJORS: the list of courses taught in each curriculum. The third sheet contains the ROOMS data: name, maximum capacity, location, and furniture type. We made the assumption that all courses require: a projector or a screen and facilities for disabled people. Finally, the last sheet contains the PERIODS-ROOMS timetable which includes the "forbidden Periods" of the timetable. All data sheets are available on the Google Drive link: [Click here](#).

	LLN-T1	LLN-T2	LLN-T3	LLN-T4	MONS-T1	MONS-T2	MONS-T3	MONS-T4
#MAJORS	19	15	6	12	9	14	7	10
#COURSES	23	42	14	37	26	41	22	31
#LESSONS	57	126	42	111	56	101	40	66
#TEACHERS	24	44	11	38	25	37	21	25

Figure 1: Overview of the main data of trimesters : highest and lowest values are respectively shown in blue and yellow.

4 Problem Formulation, Constraints and Cost function

4.1 Objectives

The LSM's University Timetable is a curriculum-based problem, meaning that *conflicts between courses are set according to the curricula published by the university and not on basis of the enrolment data*" as described for the Second International Timetabling Competition (ITC-2007) (Di Gaspero & al., 2007). Our timetable is be divided into eight independent sub-problems of that kind, one for each trimester of each campus. It should be noted that if a course is taught during two successive trimesters (typically, the bachelor's courses), it is added to both lists of courses assigned to each trimester. For each sub-problem, the LSM problem can be characterised as follows:

- A set of d teaching days (5 in our timetable).
- A set of t timeslots (4 per day).
- A set of p periods such as $p = n * d$ (20 available periods).
- A set of r rooms featured by a *Max Capacity*, a *Town District* and a *Furniture Type*.
- A set of n courses departed in a given *Number of Lessons* and attended by a given *Number of Students*.
- A set of l lessons to be scheduled in a *period* and in a *room*.
- A set of s teachers charged to teach one or more courses.
- A set of m majors composed of courses followed by a same group of students.

4.2 Hard and Soft Constraints

A good solution to the problem is one that respects certain conditions even though they may be hierarchised by importance. Common sense prioritises a timetable without course conflict over a comfortable daily schedule. This a good example of what is a hard constraint and a soft constraint. A candidate timetable violating a hard constraint cannot be accepted while the violation of one or several soft constraints may be accepted as long as the candidate solution is better than the previous one (Rossi-Doria & al., 2003). Hard constraints must be satisfied. Each candidate solution will be evaluated by the sum of hard and soft constraints violations. To do so, each constraint violation will be given a marginal cost, increasing proportionally with the weight of the respective constraint in the quality of the solution timetable (Alvarez-Valdes & al., 2001; Soria-Alcaraz & al., 2015).

The **hard constraints** that will be considered are:

- Every lesson must be scheduled and assigned to distinct periods.
- No room can be assigned to two simultaneous lessons.
- Two lessons of courses of a same major cannot be scheduled at the same period.
- No teacher can give simultaneous lessons.
- The number of students attending a course cannot exceed the maximum capacity of the room.
- A room assigned to a lesson must meet the "furniture type" requirement of the lesson.
- The LSM timetable must respect the use of rooms occupation by the other schools of the university. Not all rooms are available at every period.

The **soft constraints** that will improve the quality of the timetable are:

- Non-compactness: daily schedules should be as compact as possible and avoid "blanks" between two lessons.
- Early short morning course: if a single lesson is given in the morning, the timeslot at 10h45 is preferred to the one of 8h30.
- Long course: the length of a course should not exceed 4 hours(2 successive lessons).
- Two lessons in the Morning: favour block of 4 hours of the same course during the morning.
- Late course: Lessons after 16h15 should be avoided.
- Split course: if several lessons of the same course occur during the same day, they should not be split by another course or a blank timeslot.
- Split course by lunch: it is preferable that a course occurs before or after the lunch period but not on both sides.
- Short day: A schedule should avoid days with only one timeslot used.
- Long day: A schedule should not exceed three timeslots a day.
- Many days: The number of days used in the week should be proportional to the number of lessons existing in a major.
- Room Change during a course: room change between successive lessons of a same course should be avoided.
- Long distance between courses: the distance between two successive courses should be minimised.
- District: In LLN, rooms located close to the LSM (Centre) should be favoured.
- Low Occupancy Rate: rooms of more than 100 seats should be filled with a minimum rate of 30%

The cost function is defined as the sum of all violations of hard and soft constraints. The objective of the algorithm detailed hereafter will be to minimise this cost function. The marginal costs are given on Figure 2. The value given to each violation is relatively subjective, but the main idea here is to set a hierarchy among the constraints in three main classes: the first with the hard constraints weighted with a pseudo-infinite value; the second for important soft constraints mainly about the quality of schedules (200-600); and finally, the soft constraints with lower importance, especially about room location and comfort of students (50-100).

Cost Function's Components	Marginal Cost
HARD_Course_Conflict (Teachers and Majors)	1000000
HARD_Room_Conflict	1000000
HARD_Non_Suitable_Rooms	1000000
Cost_Non_Compactness	100
Cost_Early_Short_Morning_Course	100
Cost_Long_Course	200
Cost_2_Courses_In_Morning	50
Cost_Late_Course	100
Cost_Split_Course	600
Cost_Split_Course_By_Lunch	50
Cost_Short_Days	200
Cost_Long_Days	400
Cost_Many_Days	500
Cost_Room_Change_During_A_Course	500/50 if lunch in between
Cost_Long_Distance_Between_Courses	0/100/200/300
Cost_District	0/50/50/100
Cost_Low_Occupancy_Rate for large rooms	50

Figure 2: Components of the cost function and marginal costs

5 Solution Method

For concepts explained in the following section, the source code of realised implementations is available on the Google Drive link: [Click here](#). Although the global understanding of this thesis does not require reading them, they are at the disposal of the interested reader.

5.1 Resolution Strategy

In order to find a good solution to the LSM timetabling problem, we will implement a hybrid approach. Phase 1 will be to construct an initial feasible timetable e.g strictly respecting all hard constraints. To do so, a sequential heuristic using graph colouring and maximum matching algorithms will be implemented. Graph Colouring will help assign lessons to periods while Maximum Matching will attempt to attribute assigned lessons to a convenient room (Kostuch, 2004). Then, we attempt in Phase 2 to improve the timetable cost with a metaheuristic method: an Iterated Local Search (ILS).

5.2 The Iterated Local Search Algorithm (ILS)

The Iterated Local Search is a simple but powerful metaheuristic algorithm and the most general scheme among the explorative strategies. (Blum & Roli, 2005). This section explains the main features of this method.

After an **initial Solution** is generated, a **Local search** is applied on it until it reaches a local optimum. The *Local Search* is assumed to be a deterministic and memory-less procedure: if C is the cost function to minimise, S a set of solutions and s a candidate solution such that ($s \in S$), then for any input s , it will always output a local optimal solution s^* such that $C(s^*) \leq C(s)$. Local Search is then defined as a Many to One mapping from the set S to a smaller set S^* of local optimal solutions. A "basin of attraction" of a local minimum s^* is defined as the set of s that are mapped to s^* under the Local Search heuristic (Lourenço & al., 2003).

On the cost function C depicted on Figure [3](#), local optimal solutions $s^* \in S^*$ are located at the local minima of the curve, which are themselves located at the bottom of each basin of attraction. A candidate solution s within a basin will "slide" until its bottom.

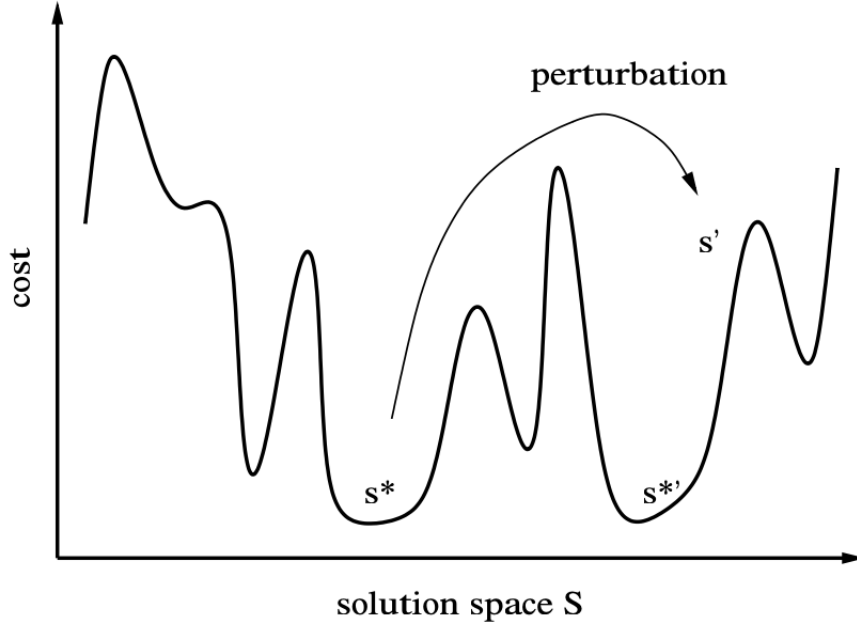


Figure 3: Pictorial representation of Iterated local search. Starting with a local minimum s^* , we apply a perturbation leading to a solution s' . After applying *LocalSearch*, we find a new local minimum $s^{*'}$ that may be better than s^* . (Lourenço & al., 2003)

The challenge now is to explore S^* in order to find a path to the global minimum of the solution space. To do so, a **Perturbation** is applied with the objective to escape from the current basin of attraction. If s^* is the local minimum reached after applying the *Local Search* on an Initial Solution s , the perturbation will lead to a new perturbed solution ($s' \in S$). Finally, a *Local Search* is one more time applied on s' to converge to the a new optimised local solution ($s^{*'} \in S^*$). If the candidate solution reached fit with an **Acceptation Criteria** (usually if $s^{*'} < s^*$), $s^{*'}$ becomes the next element of the path leading to a global minimum. The next perturbation will be applied on this last. This algorithm is looped until a certain **Stopping Criterion** is met: it may be for instance a maximum CPU time or a fixed number of iterations (Lourenço & al. 2003).

The generic algorithm of ILS is available on Figure [4](#)

5.3 Phase 1: Generate the Initial Solution

The first step of an Iterated Local Search is to generate an initial solution. But the question now is to determine which one. We could think either about a random solution,

```

 $s_0 \leftarrow \text{GenerateInitialSolution}()$ 
 $\hat{s} \leftarrow \text{LocalSearch}(s_0)$ 
while termination conditions not met do
   $s' \leftarrow \text{Perturbation}(\hat{s}, \text{history})$ 
   $\hat{s}' \leftarrow \text{LocalSearch}(s')$ 
   $\hat{s} \leftarrow \text{ApplyAcceptanceCriterion}(\hat{s}, \hat{s}', \text{history})$ 
endwhile

```

Figure 4: High level algorithm of ILS (Blum & Roli, 2005)

either a solution given by a greedy construction heuristic. As compared in Lourenço's works, the second option had two advantages over random solutions. First, a greedy initial solution combined with an ILS often converges to initial solutions s_0^* of better quality. Secondly, a local search using solutions output from a greedy algorithm take on average less improvement steps and so less computation time (Lourenço & al., 2003). This the reason why a greedy construction will be chosen to solve the LSM timetable problem.

We will use the method used by Kostuch (2004) to generate an initial solution that strictly respects hard constraints. Furthermore, some parameters can be set at the beginning of the *Generate_Initial_Solution* algorithm in order to influence the quality of the initial timetable and already reduce some soft constraints violations. As an example, we will avoid scheduling lessons on the 4th timeslot of the day (16:15-18:15) since that kind of allocation is penalised by the cost function.

5.3.1 Step 1: Pre-processing and Data Structure

In order to obtain competitive results in a convenient computation time, having efficient data structures and internal representation is essential (Kostuch, 2004). The objective of this first step is to reduce the diversified sources of information into a few compact structures.

First, about the hard constraints: two matrices will be necessary to take them into account in the model. The first one is a *lessons-room* matrix filled with 0 and 1 indicating which room is suitable for each lesson. The first matrix takes into account the maximum capacity and the type of furniture needed for each lesson. Then, the second matrix is an *incidence* matrix also filled with 0 and 1, of size (*lessons*×*lessons*) indicating for each

lessons all other lessons that cannot enter into conflict with it. Majors' constraints and teachers' constraints are included in this matrix, meaning that if a teacher has two sets of lessons of two different courses to give, those two sets of lessons must not enter into conflict. Similarly, the set of lessons composing a major cannot enter into conflict. Moreover, we do not forget the *periods* $\times$ *rooms* solution timetable containing the forbidden slots (labelled "OUT") on which lessons cannot be attributed. The only available slots of that matrix are filled with a "0".

Besides those two static matrices, we are going to define a set of dynamic data structure do to evaluate the costs of the timetable during the execution (Kostuch, 2004). Those matrices will also be very useful to visualise the schedule of each major once a solution will be generated. We define 4 different matrices:

- A (*Periods* $\times$ *Majors*) matrix whose elements are **lessons**.
- A (*Periods* $\times$ *Majors*) matrix whose elements are **rooms**.
- A (*Periods* $\times$ *Majors*) matrix whose elements are **town districts**.
- A (*Days* $\times$ *Majors*) matrix whose elements are the **number of lessons** given on each day.

Thanks to those data structures, one will be able to define all procedures that will efficiently evaluate each sub cost that composes the cost function C (See section [4.2](#)).

5.3.2 Step 2: Attribution of lessons to periods

The next step consists of the attribution of lessons to periods, such that courses-related hard constraints will be respected (majors and teachers). To do so, we go on with the strategy of Kostuch by using a concept of the Graph Theory: the **Graph Colouring algorithm**. This method will provide a better initial solution than a randomly generated one. (Sultan & al., 2008)

The graph colouring theory applied to the University Course Timetable Problem is described as follows. A timetable is a non-oriented graph $G = (V, E)$ where vertices V are the lessons and where an edge E linked to two neighbours vertices n_i, n_j indicate that n_i, n_j are incident and thus, cannot be scheduled at the same period in the timetable. In the graph colouring algorithm (or Greedy Colouring) a colour is assigned to each node

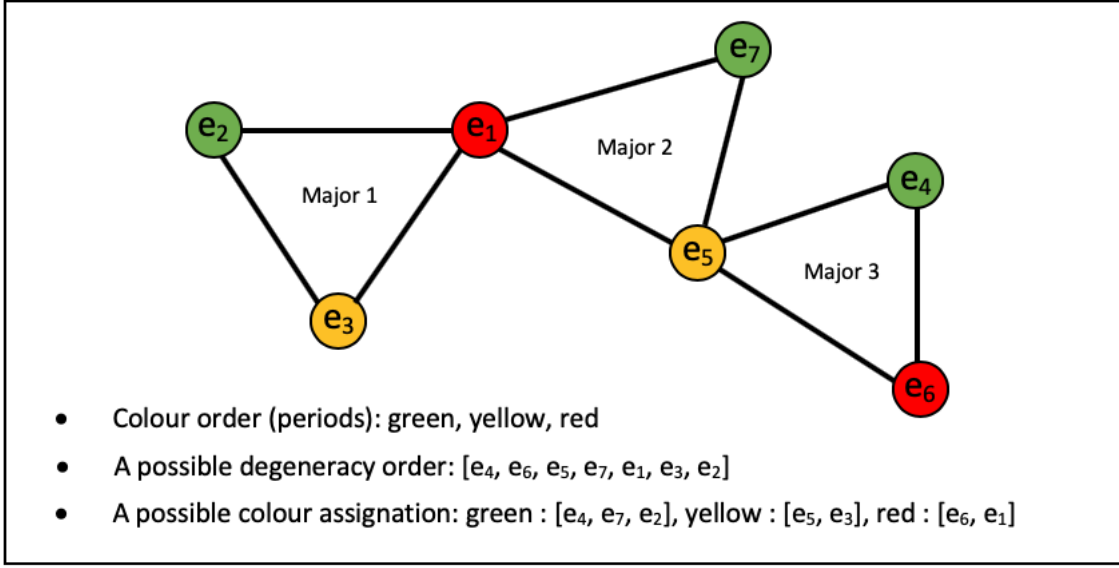


Figure 5: Illustration of the Greedy Colouring algorithm on a simplified assignment problem: 3 different majors composed of 3 lessons to allocate on 3 possible periods (colours). Colours are assigned in a degeneracy order. (E.Dive, 2020)

such that two nodes connected by an edge must not be assigned the same colour. In the timetabling problem, periods are defined as colours. The Greedy Colouring is described as follow (West, 2001) :

"The greedy colouring relative to a vertex ordering $v_1, \dots, v_n$ of $V(G)$ is obtained by colouring vertices in the order $v_1, \dots, v_n$, assigning to v_i the smallest-indexed colour not already used on its lower-indexed neighbours."

The chosen vertex ordering is the **degeneracy order**: we first take the lesson with the minimum degree in the initial graph and put it last in the list. Then, this lesson is removed from the graph. Again, we identify in the remaining subgraph the node with the minimum degree and place it in the second-to-last position of the degeneracy order, and finally remove it from the graph. We repeat this procedure until all events are placed in the ordered list (Kostuch, 2004). The Greedy Colouring algorithm is illustrated on Figure 5. Once the algorithm completed, a set of lists are output, each one related to each of the 20 available periods. Elements of each list are a set of lessons that will be scheduled on the respective candidate period.

Within the algorithm, a priority order of periods (colours) has been established in order to favour the attribution of periods respecting more the soft-constraints. Depending

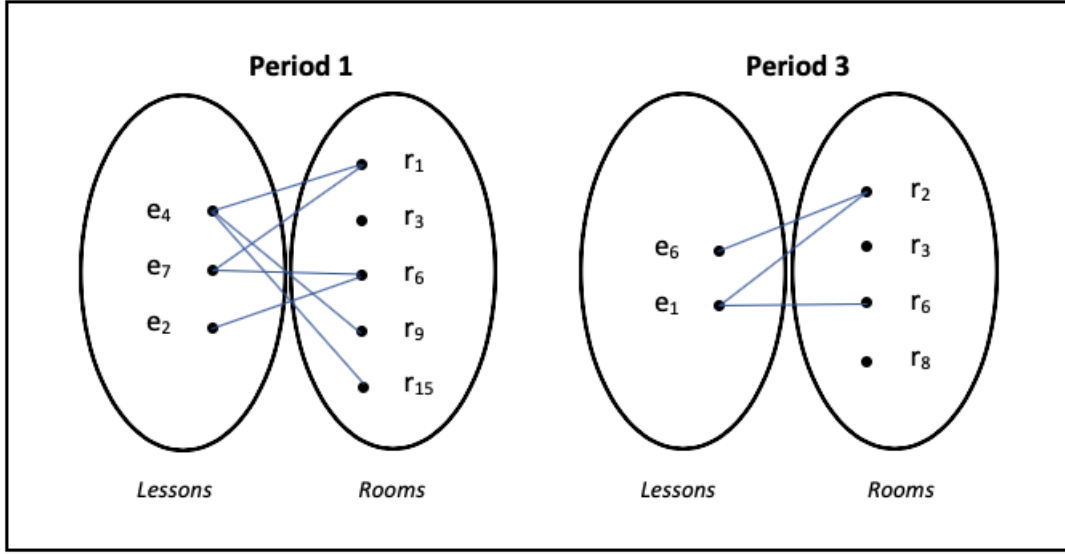


Figure 6: Illustration of two bipartite graphs generated after the Greedy Colouring algorithm. Available rooms are linked to each lesson assigned in the respective period. (E.Dive, 2020)

the expectations of the LSM secretaries and students about the final timetable, an ideal order of days used had been defined (Wednesday of Trimester 2 should be reserved for academic activities for example, the algorithm will first try to attribute the lessons to the other days).

5.3.3 Step 3: Allocation to available rooms

Once lessons are being given a candidate period, a classroom has to be attributed to each one. To do so, we run a **Maximum Matching algorithm for bipartite graphs** for each period (Kostuch, 2004). For each period, we need generate a list of all rooms available and we also take list of lessons generated by the Greedy Colouring. the left nodes of the bipartite graph are the lessons while the right nodes are the available rooms, as illustrated on Figure 6. Then, we want to indicate for each lesson which rooms is suitable in order to respect the constraints given by the *lessons-rooms* matrix (maximum capacity, furniture). We add edges between the lesson and all suitable room. Note that we actually allowed only rooms that are available during minimum two successive timeslots. This will push our model to generate an initial timetable of better quality since two lessons of a same course are preferred to be taught in a block of 4 hours as mentioned in the soft constraints.

Once the bipartite graph implemented, we use an **Augmenting Path algorithm** that will choose a room for each lesson (Ge, 2017; Papadimitriou & al, 1998). When a

suitable room is found, the lesson is added to the *periods-rooms* solution timetable. This algorithm is run for each period until all lessons are managed. However, some lessons may remain unassigned at the end of this procedure. Indeed, it is possible that no any suitable room is available for some lessons at their initial period. Hence, these lessons are placed in a pool of unassigned elements to be handled later.

5.3.4 Step 4: Management of eventual remaining lessons

The above strategy generally provides good results to assign lessons in the initial timetable. However, if the level of constraints applied on a trimester is particularly high, some events may remain unassigned (lessons strongly incident to others, few available rooms in a given day, etc.) In order to assign the remaining lessons unmanaged by the Maximum Matching, we proceed as follow: for each lesson unassigned, we generate the list of periods used by others incident lessons. These periods will be forbidden. Then, we give to that lesson a remaining available period, and finally a suitable room available on that period. We repeat until all remaining lessons are scheduled.

Although the Graph Colouring algorithm applied to the timetabling problem combined with a maximum matching algorithm provides good results to generate a timetable respecting hard-constraints, its performance remains limited. Indeed, we will see that the cost value of the initial timetable could be better. As it has been already studied before, a metaheuristic approach need to be combined with that graph colouring theory in order to improve its performance (Babaei & al., 2015). This is why the implementation of an Iterated Local Search is justified.

5.4 Phase 2: Optimisation with an Iterated Local Search (ILS)

5.4.1 Local Search

Now we have an initial solution, the challenge is to improve its quality. This solution is only one among many others within a solution space non-enumerable in reasonable time (Marmion, 2011). The Local Search is a heuristic which explores the *neighbourhood* of the current solution considered. The neighbourhood $N(s)$ of a candidate solution ($s \in S$) is defined by *all candidate solutions that can be obtained by applying specific modifications or "movements" to s* . (Stützle & Ruiz, 2018). In our problem, the Local Search consists of a succession of "swaps" of two lessons followed by an evaluation of the solution obtained

(a neighbour of the initial solution). That Local Search stage that examines neighbours of a solution is also referred as an *intensification* process (Glover & Laguna, 1998).

Among exploration methods, descending methods are quick and commonly used. Those methods search for another solution s' in the neighbourhood $N(s)$ such that if $f(s') \leq f(s)$ then replace s by s' until for all neighbour s' of s , $f(s') \leq f(s)$. When it happens, the local optimum of the neighbourhood is found. The *Hill-Climbing* method will exhaustively explore the entire neighbourhood while the *First Improvement Hill-Climbing* will stop to search when it finds a better neighbour. The *Netcrawler* method is particular because it also accepts neighbours of equal value (Marmion, 2018). The **Hill-Climbing** will be the chosen one for our model.

The question now is how to determine the limits of a candidate solution's neighbourhood, such that efficiency of the model will not be harmed. Since Hill-Climbing explores the entire neighbourhood and may be time-consuming, it is crucial to choose an appropriate definition of it. Indeed, in the largest sub-problem (LLN-Trimester 2), with 126 lessons, 15 majors, 160 rooms and 20 possible periods, the number of combinations is considerable. We must find a trade-off between an inefficient time-consuming "Global Search" exploring the entire solution space versus an ineffective "Too Small Search" not providing any sufficient improvement.

We define the neighbourhood of a current solution as follows: we first need to take a look at the weekly schedule of each major. Each major uses a certain set of rooms and a certain set of periods (where lessons have been attributed during the generation of the initial solution). A sub-matrix of the *periods-rooms* solution timetable will be formed by the set of rooms used and by the set of days in which periods used are situated. Within that restricted domain, a *Hill-Climbing* is applied: the lessons composing the considered major are swapped and all possible combinations of lessons are evaluated. Lessons can also be swapped with unused slots within the sub-matrix. For example, if the 9 lessons of the major "Supply Chain Management" are initially using 4 rooms and 9 periods among 3 different days, the *Hill-Climbing* procedure will swap lessons among all slots of the sub-matrix of size (12×4) until reaching the combination that minimises the cost of the major's schedule (as a reminder, a day is composed of 4 timeslots). This operation is applied on each major, until the local minima is found. If we look closer to that strategy, if m is the number of majors, then a single *Local Search* iteration is actually composed of m *Hill-Climbing*'s, each one exploring each major's neighbourhood. All together, they

converge to the local minima. This *Local Search* stage is deterministic, so every iteration will converge to the same local minimum as long as the current solution considered remains unchanged.

5.4.2 Perturbation

The previous step enables us to slide toward the bottom of the basin of attraction in which the current solution is located. However, only a small part of the solution space has been explored and the new optimised solution may not correspond to the global minimum of the timetable (Lourenço & al., 2003). The power of the Iterated Local Search meta-heuristic stays in the *perturbation* stage. As explained in section 5.2, the perturbation will enable the current solution to escape from its basin of attraction. To do so, a certain number of elements of the current solution will be moved to another place. That perturbation must be neither too strong, neither too weak. A too strong perturbation is equivalent to a random restart which has low probability to converge to a better solution, and actually annihilates efforts done to build a high-quality initial solution. A too weak perturbation will simply not enable the solution to escape from its basin of attraction (Lourenço & al., 2003). This stage is also referred to as a *diversification* process, in opposition to the *intensification* process of the Local Search (Glover & Laguna, 1998). Indeed, it is crucial that the perturbation procedure be fundamentally different from the Local Search otherwise no any improvement would be done (Lourenço & al., 2003). The perturbation is characterised by its non-deterministic nature.

The perturbation chosen for the LSM timetable will also be designed at the level of weekly schedules of majors. For each major, a certain number of lessons are removed from the timetable and added to a perturbation pool. The perturbation is thus characterised by its **Strength**, also called order. The less the strength, the less elements of the current solution are moved per major. We define the strength as a scale from 0 to 10 where a 0-strength neutralises the perturbation and where a 10-strength correspond to a total random restart. A 1-strength will remove 1/10 of lessons per major. A 10-strength will remove every lessons of majors. The results will show which level of Strength is needed.

Once the perturbation pool built, each lesson composing it is placed in a random slot of the *periods-rooms* solution timetable, which still respects the hard constraints of the model. To do so, the algorithm takes the first lesson, generates the list of available

periods for it and chooses one randomly. Once a period attributed, the list of available rooms is generated, and one of them is choose randomly. However, if no any suitable room is available on that period, then the algorithm chooses another random available period and picks a room again. This procedure is applied for each lesson of the perturbation pool until all of them are assigned to a new position. The Local Search procedure is then applied on the perturbed solution, making it converge toward the bottom of its basin of attraction, which must normally be different from the previous one.

5.4.3 Acceptance and Stopping Criteria

The **acceptance criterion** is determinant for the rest of the algorithm. The stake of that stage is to determine which of the previous solution and the perturbed-optimised solution will become the new current one. That criterion can influence the model and favours a diversification or rather an intensification (Lourenço & al., 2003). The perturbed-optimised solution is evaluated by the cost function and compared to the previous solution.

Among several acceptance criteria defined by Lourenço & al. (2003), the *BETTER* one is chosen. This criterion simply accepts the perturbed-optimised solution if this one has a better value than any other solution found before (Stützle, 1998). Since the model is built such that the previous solution is always the best solution found so far, the acceptance criterion is defined as follow. If s^* is the previous solution, and $s^{*'} the candidate solution then:$

$$BETTER(s^*, s^{*'}) = \begin{cases} s^{*'} & \text{if } C(s^{*'}) < C(s^*) \\ s^* & \text{otherwise.} \end{cases}$$

Once the ILS launched, iterations of Perturbation + Local Search will be done until a **Stopping Criterion** is reached. This criterion can be a maximum number of iterations or a maximum CPU time (Marmion, 2011).

The model has been run on FICO XPRESS Workbench whose programming language is Mosel. The computer used ran on macOS Catalina version 10.15.6, with a 2.5 GHz Intel Core i5 and 4 Gb of RAM.

6 Numerical Results and Analyses

This section first reveals the results obtained by the combination of the Greedy Algorithm and the Iterated Local Search (ILS) detailed above. The results of the ILS shown in the first subsection are based on five executions of the model with a perturbation strength equal to 1 for all eight trimesters. In total, eight sets of five instances are generated and the model stops when 15 iterations have been realised (stopping criterion). Secondly, we compare the best results obtained by the model with the LSM timetable of the academic year 2019-2020 in order to evaluate the effectiveness of such a model. Finally, the impact of the variation of the perturbation strength is discussed in the third subsection.

6.1 Results of the ILS

Global results are depicted on Figure 7 and Figure 8. After having generated an initial solution in negligible CPU time, we can figure out that the ILS provides a very satisfactory solution for each trimester. Indeed, after 15 iterations only, the cost values of the eight trimester timetables sharply fall with an average improvement rate located between -48% and -86%. The global average of the entire LSM timetable cost plunges with an average of -69 %, and the best global solution found reaches a peak of -72%. Over the five instances generated, the standard deviation converted in percentage of the initial solution is limited to 3%.

	RunTime First Local Search (min)	Total RunTime (min)	Greedy Initial Solution	After First Local Search	Best Solution Found after 15 iterations	Average Minimum Reached (μ)	Standard deviation (σ)/Initial Solution
LLN-T1	1,8	39,0	22850	14050	11300	11950	3%
LLN-T2	11,5	141,0	17650	8050	5300	5600	2%
LLN-T3	0,4	17,0	4600	3600	1250	1990	15%
LLN-T4	9,5	123,0	13000	4500	2300	2720	2%
MONS-T1	0,6	8,0	11250	2850	1400	1560	1%
MONS-T2	0,9	15,0	14300	3150	1950	2310	3%
MONS-T3	0,4	4,0	7250	3450	2600	2660	1%
MONS-T4	0,5	11,0	11250	3250	2450	2710	2%
LSM-TOTAL	25,6	358,0	102150	42900	28550	31500	3%

Figure 7: Results obtained after having run 5 instances of each trimester with a stopping criterion of 15 iterations. Detailed costs are available in Annexes.

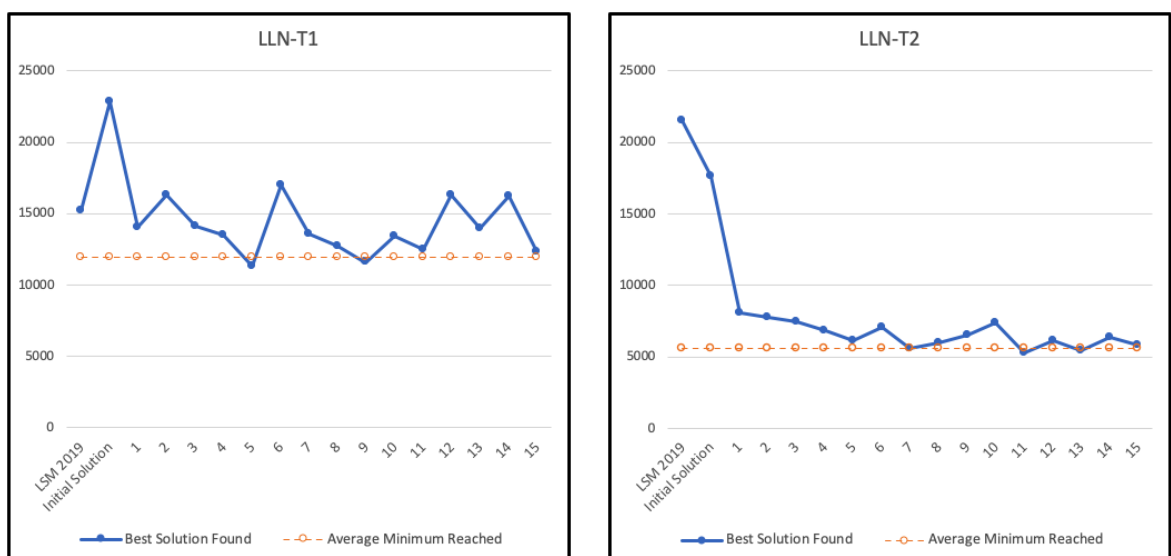
In addition, we can observe that an important part of that cost diminution occurs during the first *Local Search*, the one optimising the initial solution. During that first step,

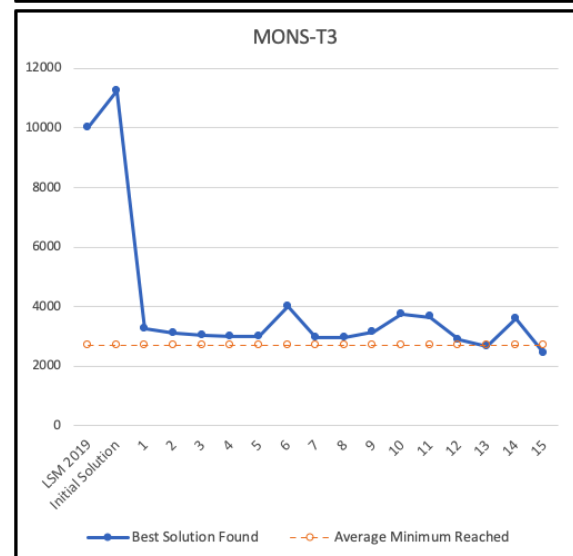
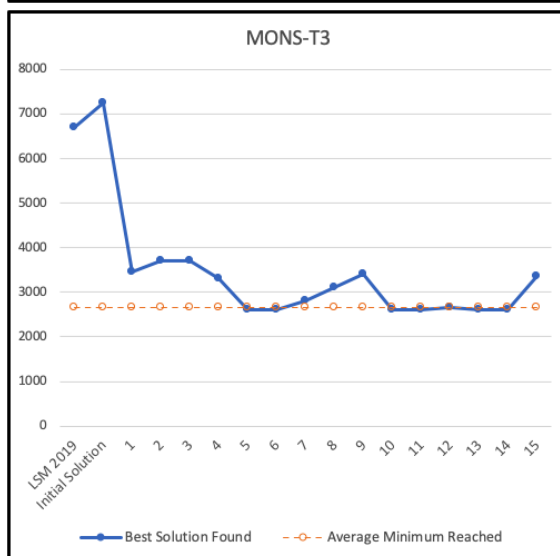
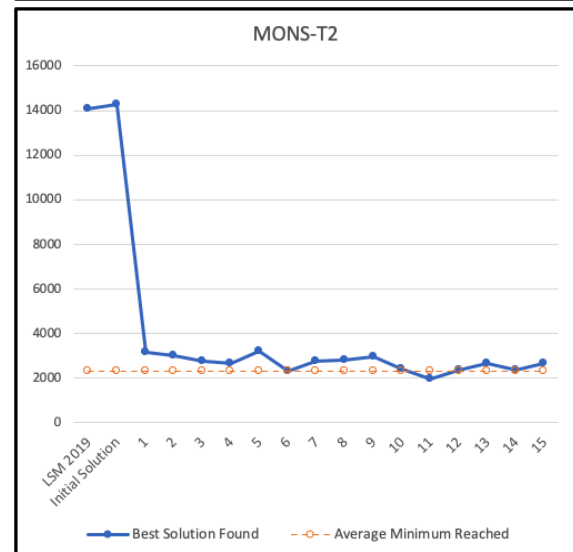
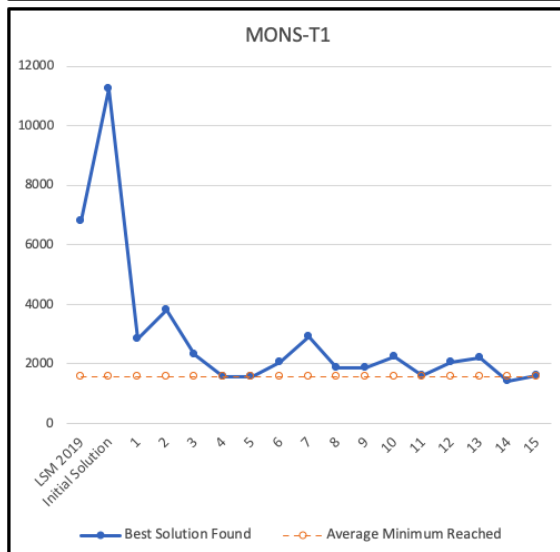
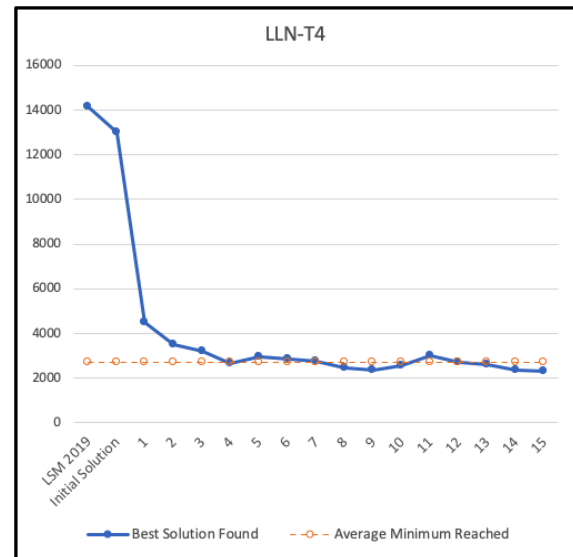
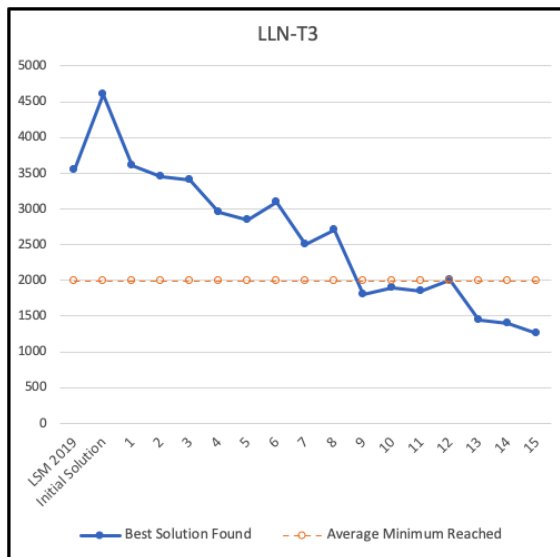
	Improvement after first Local Search	Best Improvement after 15 iterations	Average Improvement after 15 iterations
LLN-T1	-39%	-51%	-48%
LLN-T2	-54%	-70%	-68%
LLN-T3	-22%	-73%	-57%
LLN-T4	-65%	-82%	-79%
MONS-T1	-75%	-88%	-86%
MONS-T2	-78%	-86%	-84%
MONS-T3	-52%	-64%	-63%
MONS-T4	-71%	-78%	-76%
LSM-TOTAL	-58%	-72%	-69%

Figure 8: Best and average improvements in % of the Initial Solutions.

the cost functions of each timetable dramatically collapse for seven out eight trimesters. Afterward, the curves usually slightly but surely decline along the successive iterations. The first *Local Search*, in a deterministic way, is responsible of a decrease situated between -39% and -78% of the initial solution value of these seven trimesters. However, the curve of LLN-T3 shows a less steep slope, although it reaches at the end the same level of average improvement. An important strength of this model is that it only provides feasible solution, e.g all hard constraints are respected.

The best solutions reached for each trimester can be seen on the eight charts hereafter. The improvement of the cost value along the iterations can be seen. The orange line indicates the average minimum reached after the five iterations.





Furthermore, we can observe that the instances have been generated in very reasonable CPU times. After 25 minutes, an average improved solution of -58% is found after the first *Local Search*, with a cost value of 42.900, vs 102.105 initially (See Figure 7). Realising 15 iterations takes six hours, but this is mainly due to two specific timetables. Indeed, CPU time increases proportionally with the number of lessons and more specifically with the size of majors. One can see that each iteration of timetables LLN-T2 and LLN-T4 takes in average 10 minutes to be done, while other trimesters roughly need 1 minute. This is explained by the largest number of lessons to schedule in many majors of these two trimesters. Since the number of lessons is larger, the number of *Swaps* realised by the *Local Search* is larger too. If a major has more lessons to schedule, the probability that this major uses more rooms and more days is bigger too, which has an influence on the size of the domain where *Swaps* are realised. The CPU time used by the *Hill-Climbing* method increases quadratically with the size of the domain (the sub-matrix of the *periods-rooms* solution timetable). This is probably an important weakness of this model. In worse conditions, the CPU time could increase a lot.

6.2 Comparison with the real LSM-2019 timetable

In order to assess the effectiveness and usefulness of our model, we compare the solution found with the real timetable used by the LSM during the 2019-2020 academic year. To do so, the entire LSM-2019 timetable has been scanned and evaluated by our cost function. The values obtained can be found on Figure 9 and all cost details can be consulted in ANNEXE 10.1. These tables give us the opportunity to compare our solution.

As it can be seen on Figure 10, the initial solution generated provides already a feasible solution not so far away from the LSM-2019 value (92.100 vs 102.150). Then, the cost value of our solution considerably decreases to 42.900 during the first *Local Search*, surpassing by far the LSM-2019 value. Finally, the curve keeps on going down, but in a lesser extent though. After 15 iterations, the value of 28.550 is reached by the best instance of each trimester, corresponding to improvement of -69% in comparison with the LSM-2019 timetable.

If we inspect at the level of trimesters, the solutions proposed are clearly better for all eight trimesters than the LSM-2019 timetable. However, we can see that the quality of the best solution found for Trimester LLN-T1 is less than the others. Two possible causes can

	LSM Timetable 2019	Greedy Initial Solution	After First Local Search	Best Solution Found after 15 Iterations	Average Minimum Reached (μ)	Best Improvement of LSM 2019	Average Improvement after 15 iterations
LLN-T1	15250	22850	14050	11300	11950	-26%	-22%
LLN-T2	21550	17650	8050	5300	5600	-75%	-74%
LLN-T3	3550	4600	3600	1250	1990	-65%	-44%
LLN-T4	14150	13000	4500	2300	2720	-84%	-81%
MONS-T1	6800	11250	2850	1400	1560	-79%	-77%
MONS-T2	14100	14300	3150	1950	2310	-86%	-84%
MONS-T3	6700	7250	3450	2600	2660	-61%	-60%
MONS-T4	10000	11250	3250	2450	2710	-76%	-73%
LSM-TOTAL	92100	102150	42900	28550	31500	-69%	-66%

Figure 9: Comparison of the best results obtained for each trimester with the LSM timetable of 2019. Comparison done after 15 iterations.

be raised. A first explanation could be a lack of chance in the iterations generated for this trimester. Since only five of them have been realised, the non-deterministic characteristic of the ILS could provide five poor solutions. Another explanation could be the difficulty for the model to find a better solution due to the level of constraints. Indeed, Trimester LLN-T1 in particular is characterised by a high level of constraints. This semester has an important number of majors, for a relatively few number of courses. Several courses are also taught by the same teachers, which has an important impact on the incidence of lessons.

Under the set of constraints defined, it can be concluded that the model proposes a global effective solution to the LSM timetable problem. Those good and encouraging results must nonetheless be taken with caution as it will be explained in the critical analysis. An example of best-found timetable is in ANNEXE [10.2](#). The other ones are available on the Google Drive ([Click here](#)).

6.3 Variation of the Perturbation Strength

The choice to set the perturbation strength to 1 (on a scale of 0 to 10) is not done without reason. The model has been tested for different level of strength.

It appeared that strongly constrained timetables only provided deceiving results, even for a 2-strength (for example LLN-Trimester 1, with many majors and few courses). Indeed, in strongly constrained timetables, a higher proportion of lessons are taught in

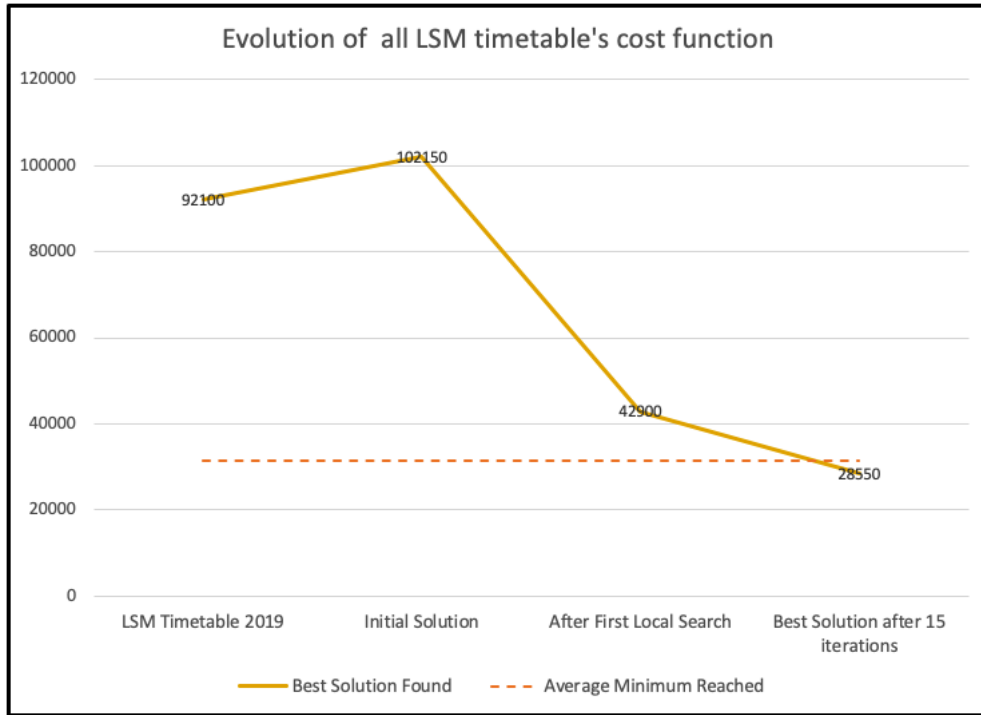


Figure 10: Evolution of the total LSM timetable's cost function. Detailed costs are available in Annexes and on Google drive. The real value of the LSM timetable used in 2019 can be compared to the solution obtained with our model.

several different majors. Let's take two majors with common lessons, Major 1 and Major 2. During the perturbation of higher strength, more lessons are removed from each major and thus from the solution timetable, by definition of the perturbation. But when a particular lesson is removed from Major 1, it is also removed from Major 2. The high dependence of the two majors combined with a strong perturbation amplify this latter. We are in that case closer from a simple random restart, which is not compatible with our model. Indeed, the probability to find a best solution with a random restart is low, and the greedy construction realised for the initial solution is neutralised. A solution would be to dramatically increase the number of iterations with the hope to reach a better cost value.

For less constrained timetables, if the perturbation strength is set to 2, the model provides similar results as for strength set to 1. However, the total CPU time is considerably increased. In that case, increasing the strength only decrease the efficiency of the model. For higher strength value, we fall in the problem of random or almost random restart.

In short, a 1-strength provides the only and best performance in term of cost value and CPU time. For this model, this is the best trade-off between diversification and intensification. For a 2-strength or a 3-strength and for low constrained timetables, the model loses in efficiency but provide similar cost values. For higher strength, the model loses all effectiveness and efficiency whatever the timetable.

7 Critical Analysis and limitations

Besides these encouraging results, it is important to keep in mind that this thesis is perfectible and that results must be put in perspective. Indeed, some assumptions that have been set up may influence the results. We first think about the teachers' preferences and availabilities that have not been taken into account since no useful information were available. We can mention the fact that the solution timetable only uses a defined number of periods and timeslots, whose length has been fixed. Moreover, the choice of soft constraints was partly based on a subjective point of view. All stakeholders concerned by the timetable (teachers, other students, employees managing the timetable) could have a different opinion on what is the ideal timetable.

In addition, the cost function probably needs to be adapted depending the academic program. Indeed, when looking at the bachelor and master's timetables, we can see that they were differently designed. Bachelors' programs contained for example a lot of exercise sessions that were not taken into account in our model. In contrast, Master's courses are more homogeneous (no exercise sessions) but more lessons are scheduled per week. In the model, the cost function has been more designed with the master's program point of view.

Furthermore, the assumption that all courses are regularly scheduled during the trimester is probably the one that make the model to deviate the most from the reality of the LSM timetable. We could imagine implementing a model that optimises a timetable not by trimester, but by week. However, such a model would require a larger CPU time.

More importantly, if results reached by the model seem way better than the LSM 2019 timetable, some bias could justify a part of it. Indeed, we must also take into account that LSM secretaries had probably other data to build the timetable, and that modifications had to be done once the first version of the timetable submitted. We only obtained the last version of it. In the model, some real constraints are not taken into account. As we

already talked about the teachers' preferences and availability, we can also mention the varying occupancy of the rooms by other schools of the university. Since the model is less constrained, a better cost value has more probability to occur. It would be potentially interesting to use the model to build the first version of the timetable and eventually apply some modifications after receiving all stakeholders' opinions about it, until the obtention of a model that respect the reality.

Another reason that could explain the large gap between the real and obtained timetable: some little data adaptations had to be done in order to make possible a comparison. For example, when uniformising irregular courses (lessons after 18:00 or on Saturday, variable weekly schedules, etc.). Modifications were done in a way it respected the most the soft constraints in order not to interfere too much with the final cost value of the LSM-2019 timetable.

Additionally, statistic data provided for the non-determinist part of the model and thus numerical results must be taken with caution since the number of instances (5 per timetable) is relatively low. An error margin exists on these data. However, the first results obtained show a promising trend, that should be confirmed with more instances.

Finally, the model could be improved in terms of programming efficiency. Since it was not the main purpose of this thesis, the source code may be designed in a non-efficient way. That could actually be an important weakness of the model. The CPU time used to finish an iteration is mainly depending of the time used by the *Local Search* when realising an exhaustive exploration of neighbourhood (*Hill-Climbing*). Optimising this part or finding another search algorithm may be necessary for harder problems. However, this issue remained relatively limited in the LSM timetable which actually produced satisfying results, but applying the Iterated Local Search strategy with an intuitive code could become problematic when applied to more demanding problem asking for a solution in a very short CPU time.

8 Conclusions

The aim of that Master's thesis was to tackle the University Course Timetabling Problem of the LSM and to prove that a better solution than the hand-made method was possible with an automated timetable using heuristics. After having explored the literature about approached solutions of the UCTP problem, a resolution strategy of the LSM timetable based on a hybridised approach has been designed. The first stage included the use of a greedy construction algorithm based on the Graph Theory to build a feasible initial solution in short CPU time and of better quality than the solution generated by a random construction. The second stage concerned the implementation of an Iterated Local Search which has provided satisfactory and competitive results. Indeed, it has been showed that the quality of the timetable obtained was significantly better than the timetable used by the LSM in the 2019-2020 academic year. However, these results must be taken with caution since some assumptions done may incur a bias to the comparative analyse realised with the LSM-2019 timetable. Nevertheless, a future collaboration with the administrative instances of the university along with improvements of the model and data refinement may be an opportunity for a successful application.

To go further, one could now imagine applying such metaheuristics methods like Iterated Local Search to the tenfold existing timetabling and assignation problems of the industrial world. In general, the solutions offered by the Operational Research field give a tremendous opportunity to improve the functioning of organisations, optimise the institutions and provide to a continuously evolving society the tools at the service of its population.

I must admit that the completion of this master's thesis has brought me more than I was expecting. Beyond learning a lot about Operational Research and programming, which is a real asset, I feel that I significantly developed my soft skills: time management, confidence, autonomy, motivation, and the ability to push the limits of my knowledges and to search for rigorous solutions. I feel more confident about what I can build with my hand and the challenge to write a thesis has helped me to better know how I am functioning. I identified or confirmed my weaknesses, which arose inexorably during the first months. But I also discovered what gave me the strength and the motivation to overcome the obstacles and cross the finish line. Today, I am more confident, ready to pursue my professional career and more equipped to face future challenges.

References

- Abdullah, S. (2006). *Heuristic approaches for university timetabling problems* (Doctoral dissertation). Citeseer.
- Abdullah, S., Burke, E. K. & McCollum, B. (2007). A hybrid evolutionary approach to the university course timetabling problem, In *2007 IEEE Congress on Evolutionary Computation*. IEEE.
- Abdullah, S., Burke, E. K. & Mccollum, B. (2005). An investigation of variable neighbourhood search for university course timetabling, In *The 2nd multidisciplinary international conference on scheduling: Theory and applications (mista)*.
- Alsmadi, O. M. K., Za'er, S., Abu-Al-Nadi, D. I. & Algsoon, A. (2011). A novel genetic algorithm technique for solving university course timetabling problems, In *International workshop on systems, signal processing and their applications, wossipa*. IEEE.
- Alvarez-Valdes, R., Crespo, E. & Tamarit, J. M. (2002). Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research*, 137(3), 512–523.
- Aycaan, E. & Ayav, T. (2009). Solving the course scheduling problem using simulated annealing, In *2009 IEEE International Advance Computing Conference*. IEEE.
- Babaei, H., Karimpour, J. & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86, 43–59.
- Bakır, M. A. & Aksop, C. (2008). A 0-1 integer programming approach to a university timetabling problem. *Hacettepe Journal of Mathematics and Statistics*, 37(1), 41–55.
- Blum, C., Correia, S., Dorigo, M., Paechter, B., Rossi-Doria, O. & Snoek, M. (2002). A ga evolving instructions for a timetable builder, In *4th international conference on practice and theory of automated timetabling, patat 2002*. KaHo Sint-Lieven.
- Blum, C. & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3), 268–308.
- Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12), 1695–1724.
- Daskalaki, S., Birbas, T. & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153(1), 117–135.

- de Werra, D. (1985). An introduction to timetabling. *European journal of operational research*, 19(2), 151–162.
- Di Gaspero, L., McCollum, B. & Schaerf, A. (2007). *The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3)* (tech. rep.). Citeseer.
- Dimopoulou, M. & Miliotis, P. (2001). Implementation of a university course and examination timetabling system. *European Journal of Operational Research*, 130(1), 202–213.
- Dorigo, M., Birattari, M. & Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4), 28–39.
- Dorigo, M. & Stützle, T. (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances, In *Handbook of metaheuristics*. Springer.
- Focacci, F., Laburthe, F. & Lodi, A. (2003). Local search and constraint programming, In *Handbook of metaheuristics*. Springer.
- Ge, R. (2017, 31th October). Design and analysis of algorithms : Bipartite matching [slides]. *Duke.edu*. <https://www2.cs.duke.edu/courses/fall17/compsci330/lecture16note.pdf>
- Glover, F. & Laguna, M. (1998). Tabu search, In *Handbook of combinatorial optimization*. Springer.
- Gotlib, C. C. (1963). The construction of class-teacher timetables. *Proceedings of IFIP Congress*, 62, 73–77.
- Kostuch, P. (2004). The university course timetabling problem with a three-phase approach, In *International conference on the practice and theory of automated timetabling*. Springer.
- Lewis, R. & Paechter, B. (2005). Application of the grouping genetic algorithm to university course timetabling, In *European conference on evolutionary computation in combinatorial optimization*. Springer.
- Lourengo, H. R., Martin, O. C. & Stützle, T. (2003). Iterated local search, In *Handbook of metaheuristics*. Springer.
- Marmion, M.-É. (2011). *Recherche locale et optimisation combinatoire: De l'analyse structurelle d'un problème à la conception d'algorithmes efficaces* (Doctoral dissertation).
- Papadimitriou, C. H. & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Courier Corporation.
- Petrovic, S. & Burke, E. K. (2004). University timetabling.

- Razak, H. A., Ibrahim, Z. & Hussin, N. M. (2010). Bipartite graph edge coloring approach to course timetabling, In *2010 international conference on information retrieval & knowledge management (camp)*. IEEE.
- Rich, D. C. (1995). A smart genetic algorithm for university timetabling, In *International conference on the practice and theory of automated timetabling*. Springer.
- Rossi-Doria, O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L. M., Knowles, J., Manfrin, M., Mastrolilli, M., Paechter, B. Et al. (2002). A comparison of the performance of different metaheuristics on the timetabling problem, In *International conference on the practice and theory of automated timetabling*. Springer.
- Schaerf, A. (1999). A survey of automated timetabling. *Artificial intelligence review*, 13(2), 87–127.
- Selim, S. M. (1988). Split vertices in vertex colouring and their application in developing a solution to the faculty timetable problem. *The Computer Journal*, 31(1), 76–82.
- Socha, K., Knowles, J. & Sampels, M. (2002). A max-min ant system for the university course timetabling problem, In *International workshop on ant algorithms*. Springer.
- Song, T., Liu, S., Tang, X., Peng, X. & Chen, M. (2018). An iterated local search algorithm for the university course timetabling problem. *Applied Soft Computing*, 68, 597–608.
- Soria-Alcaraz, J. A., Özcan, E., Swan, J., Kendall, G. & Carpio, M. (2016). Iterated local search using an add and delete hyper-heuristic for university course timetabling. *Applied Soft Computing*, 40, 581–593.
- Stützle, T. (1998). Applying iterated local search to the permutation flow shop problem.
- Stützle, T. & Ruiz, R. (2018). Iterated local search, In *Handbook of heuristics*. Springer.
- Sultan, A. B. M., Mahmod, R., Sulaiman, M. N. & Bakar, M. R. A. (2008). Selecting quality initial random seed for metaheuristic approaches: A case of timetabling problem. *Int J Comput Internet Manag*, 16(1), 8.
- Tuga, M., Berretta, R. & Mendes, A. (2007). A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem, In *6th ieee/acis international conference on computer and information science (icis 2007)*. IEEE.
- Turabieh, H. & Abdullah, S. (2009). Incorporating tabu search into memetic approach for enrolment-based course timetabling problems, In *2009 2nd conference on data mining and optimization*. IEEE.
- Welsh, D. J. A. & Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10, 85–86.

- West, D. B. Et al. (1996). *Introduction to graph theory* (Vol. 2). Prentice hall Upper Saddle River, NJ.
- White, G. M. & Zhang, J. (1997). Generating complete university timetables by combining tabu search with constraint logic, In *International conference on the practice and theory of automated timetabling*. Springer.

10 Annexes

10.1 Costs details

DETAILED COSTS ALL LSM TIMETABLE						
Objective Function components	Marginal Cost	LSM 2019	Initial Solution	After First Local Search	Best solution After ILS	Best Improvement of LSM 2019
HARD_Course_Conflict (Teachers and Majors)	1000000	0	0	0	0	0
HARD_Room_Conflict	1000000	0	0	0	0	0
HARD_Non_Suitable_Rooms	1000000	0	0	0	0	0
Cost_Non_Compactness	100	1300	600	200	600	-53,8%
Cost_Early_Short_Morning_Course	100	400	1500	0	100	-75,0%
Cost_Long_Course	200	16200	32600	9600	2600	-84,0%
Cost_2_Courses_In_Morning	50	150	1650	2000	1600	966,7%
Cost_Late_Course	100	12500	1200	2000	3200	-74,4%
Cost_Split_Course	600	0	3600	0	0	0,0%
Cost_Split_Course_By_Lunch	50	50	1300	600	50	0,0%
Cost_Short_Days	200	9400	6400	1400	1400	-85,1%
Cost_Long_Days	400	20800	800	1600	400	-98,1%
Cost_Many_Days	500	21000	14500	10000	10500	-50,0%
Cost_Room_Change_During_A_Course	500/50	1950	22450	2050	300	-84,6%
Cost_Long_Distance_Between_Courses	100/200/300	200	0	0	200	0,0%
Cost_District	50/50/100	6300	12050	11100	6850	8,7%
Cost_Low_Occupancy_Rate for large rooms	50	1850	3500	1750	750	-59,5%
TOTAL COST		92100	102150	42300	28550	-69,0%

Figure 11: Summarise of cost details and comparison between LSM-2019 and the best found solution

GREEDY INITIAL SOLUTION										
Objective Function components	Marginal Cost	LLN-T1	LLN-T2	LLN-T3	LLN-T4	MONS-T1	MONS-T2	MONS-T3	MONS-T4	ALL LSM
HARD_Course_Conflict (Teachers and Majors)	1000000	0	0	0	0	0	0	0	0	0
HARD_Room_Conflict	1000000	0	0	0	0	0	0	0	0	0
HARD_Non_Suitable_Rooms	1000000	0	0	0	0	0	0	0	0	0
Cost_Non_Compactness	100	100	0	0	0	200	100	200	0	600
Cost_Early_Short_Morning_Course	100	0	0	0	0	300	200	600	400	1500
Cost_Long_Course	200	4000	8200	2800	7800	2200	5200	200	2200	32600
Cost_2_Courses_In_Morning	50	600	0	0	0	150	100	400	400	1650
Cost_Late_Course	100	700	400	0	0	0	100	0	0	1200
Cost_Split_Course	600	3600	0	0	0	0	0	0	0	3600
Cost_Split_Course_By_Lunch	50	300	0	0	0	150	100	350	400	1300
Cost_Short_Days	200	2200	1000	0	0	800	600	1000	800	6400
Cost_Long_Days	400	0	800	0	0	0	0	0	0	800
Cost_Many_Days	500	5500	1500	0	0	1500	1000	3000	2000	14500
Cost_Room_Change_During_A_Course	500/50	950	2000	450	1950	5350	6300	1200	4250	22450
Cost_Long_Distance_Between_Courses	100/200/300	0	0	0	0	0	0	0	0	0
Cost_District	50/50/100	4650	3500	1100	2800	0	0	0	0	12050
Cost_Low_Occupancy_Rate for large rooms	50	250	250	250	450	600	600	300	800	3500
TOTAL COST		22850	17650	4600	13000	11250	14300	7250	11250	102150

Figure 12: Cost details of Initial Solution of each trimester

INITIAL SOLUTION AFTER FIRST LOCAL SEARCH										
Objective Function components	Marginal Cost	LLN-T1	LLN-T2	LLN-T3	LLN-T4	MONS-T1	MONS-T2	MONS-T3	MONS-T4	ALL LSM
HARD_Course_Conflict (Teachers and Majors)	1000000	0	0	0	0	0	0	0	0	0
HARD_Room_Conflict	1000000	0	0	0	0	0	0	0	0	0
HARD_Non_Suitable_Rooms	1000000	0	0	0	0	0	0	0	0	0
Cost_Non_Compactness	100	100	100	0	0	0	0	0	0	200
Cost_Early_Short_Morning_Course	100	0	0	0	0	0	0	0	0	0
Cost_Long_Course	200	1200	3600	2000	1600	600	600	0	0	9600
Cost_2_Courses_In_Morning	50	300	100	0	50	400	300	300	550	2000
Cost_Late_Course	100	1000	300	0	0	300	400	0	0	2000
Cost_Split_Course	600	0	0	0	0	0	0	0	0	0
Cost_Split_Course_By_Lunch	50	300	0	0	0	100	100	100	0	600
Cost_Short_Days	200	1200	0	0	0	0	200	0	0	1400
Cost_Long_Days	400	1600	0	0	0	0	0	0	0	1600
Cost_Many_Days	500	3000	0	0	0	1000	1000	3000	2000	10000
Cost_Room_Change_During_A_Course	500/50	300	750	300	300	150	250	0	0	2050
Cost_Long_Distance_Between_Courses	100/200/300	0	0	0	0	0	0	0	0	0
Cost_District	50/50/100	4800	2950	1050	2300	0	0	0	0	11100
Cost_Low_Occupancy_Rate for large rooms	50	250	250	250	250	300	300	50	100	1750
TOTAL COST		14050	8050	3600	4500	2850	3150	3450	2650	42300

Figure 13: Cost details of the solution of each trimester after the first *Local Search*.

BEST SOLUTION GENERATED BY THE MODEL (15 Iterations)										
Objective Function components	Marginal Cost	LLN-T1	LLN-T2	LLN-T3	LLN-T4	MONS-T1	MONS-T2	MONS-T3	MONS-T4	ALL LSM
HARD_Course_Conflict (Teachers and Majors)	1000000	0	0	0	0	0	0	0	0	0
HARD_Room_Conflict	1000000	0	0	0	0	0	0	0	0	0
HARD_Non_Suitable_Rooms	1000000	0	0	0	0	0	0	0	0	0
Cost_Non_Compactness	100	100	300	100	0	0	100	0	0	600
Cost_Early_Short_Morning_Course	100	0	100	0	0	0	0	0	0	100
Cost_Long_Course	200	2200	200	0	0	200	0	0	0	2600
Cost_2_Courses_In_Morning	50	0	400	50	150	200	400	100	300	1600
Cost_Late_Course	100	800	1200	400	400	0	300	0	100	3200
Cost_Split_Course	600	0	0	0	0	0	0	0	0	0
Cost_Split_Course_By_Lunch	50	0	50	0	0	0	0	0	0	50
Cost_Short_Days	200	1400	0	0	0	0	0	0	0	1400
Cost_Long_Days	400	0	0	0	0	0	0	0	400	400
Cost_Many_Days	500	3500	500	500	0	1000	1000	2500	1500	10500
Cost_Room_Change_During_A_Course	500/50	250	50	0	0	0	0	0	0	300
Cost_Long_Distance_Between_Courses	100/200/300	0	200	0	0	0	0	0	0	200
Cost_District	50/50/100	2800	2300	150	1600	0	0	0	0	6850
Cost_Low_Occupancy_Rate for large rooms	50	250	0	50	150	0	150	0	150	750
TOTAL COST		11300	5300	1250	2300	1400	1950	2600	2450	28550

Figure 14: Cost details of the best solution found of each trimester

LSM TIMETABLE 2019										
Objective Function components	Marginal Cost	LLN-T1	LLN-T2	LLN-T3	LLN-T4	MONS-T1	MONS-T2	MONS-T3	MONS-T4	ALL LSM
HARD_Course_Conflict (Teachers and Majors)	1000000	0	0	0	0	0	0	0	0	0
HARD_Room_Conflict	1000000	0	0	0	0	0	0	0	0	0
HARD_Non_Suitable_Rooms	1000000	0	0	0	0	0	0	0	0	0
Cost_Non_Compactness	100	300	0	100	600	200	100	0	0	1300
Cost_Early_Short_Morning_Course	100	0	0	0	0	0	0	100	300	400
Cost_Long_Course	200	4600	5800	200	2800	200	1600	0	1000	16200
Cost_2_Courses_In_Morning	50	0	100	0	50	0	0	0	0	150
Cost_Late_Course	100	1500	2200	300	1700	1600	2400	1200	1600	12500
Cost_Split_Course	600	0	0	0	0	0	0	0	0	0
Cost_Split_Course_By_Lunch	50	50	0	0	0	0	0	0	0	50
Cost_Short_Days	200	1800	1600	600	800	800	800	1600	1400	9400
Cost_Long_Days	400	1200	2800	0	3600	2000	7200	800	3200	20800
Cost_Many_Days	500	4000	3000	2000	2500	2000	2000	3000	2500	21000
Cost_Room_Change_During_A_Course	500/50	50	1600	0	300	0	0	0	0	1950
Cost_Long_Distance_Between_Courses	100/200/300	0	100	0	100	0	0	0	0	200
Cost_District	50/50/100	1500	3300	200	1300	0	0	0	0	6300
Cost_Low_Occupancy_Rate for large rooms	50	250	1050	150	400	0	0	0	0	1850
TOTAL COST		15250	21550	3550	14150	6800	14100	6700	10000	92100

Figure 15: Cost details of real LSM-2019 timetable of each trimester

10.2 An example of best timetable found

Trimester 2 LLN - LESSONS A	CEMS	Supply Chain	Innovation Management	Financial Engineering	Marketing Strategy for Connected Brands	Corporate Sustainable Management	International Finance (LLN)	CPME
Monday_8h30	0	0	LLSMS2041	0	0	0	0	LCPME2002
Monday_10h45	0	0	LLSMS2042	0	LLSMS2007	0	0	0
Monday_14h00	0	0	LLSMS2040	0	LLSMS2000	0	0	LFSA2212
Monday_16h15	0	0	0	0	LLSMS2000	0	0	LCPME2001
Tuesday_8h30	LLSMS2109	LLSMS2030	0	LLSMS2224	0	LLSMS2281	LLSMS2027	0
Tuesday_10h45	LLSMS2109	LLSMS2030	0	LLSMS2224	0	LLSMS2282	LLSMS2027	LCPME2002
Tuesday_14h00	LLSMS2111	LLSMS2035	0	LLSMS2225	0	0	LLSMS2054	LLSMX2001
Tuesday_16h15	0	0	0	0	0	LLSMS2280	0	LLSMS2014
Wednesday_8h30	0	0	LLSMS2041	0	0	LLSMS2281	0	LCPME2002
Wednesday_10h45	0	0	LLSMS2041	0	LLSMS2000	LLSMS2281	0	LLSMS2014
Wednesday_14h00	0	0	LLSMS2042	0	LLSMS2002	LLSMS2282	0	LLSMX2001
Wednesday_16h15	0	0	0	0	LLSMS2002	0	0	0
Thursday_8h30	LCEMS2108	LLSMS2035	0	LLSMS2013	0	LLSMS2280	0	LFSA2212
Thursday_10h45	LCEMS2108	LLSMS2035	0	LLSMS2013	0	LLSMS2280	LLSMS2054	LLSMX2001
Thursday_14h00	LLSMS2109	LLSMS2031	0	LLSMS2224	0	LLSMS2282	LLSMS2027	LCPME2001
Thursday_16h15	0	0	0	0	0	0	LLSMS2223	0
Friday_8h30	LLSMS2111	LLSMS2031	LLSMS2040	LLSMS2225	0	0	LLSMS2223	LLSMS2014
Friday_10h45	LLSMS2111	LLSMS2031	LLSMS2040	LLSMS2225	LLSMS2002	0	LLSMS2223	LCPME2001
Friday_14h00	LCEMS2108	LLSMS2030	LLSMS2042	LLSMS2013	LLSMS2007	0	LLSMS2054	LFSA2212
Friday_16h15	0	0	0	0	LLSMS2007	0	0	0

Trimester 2 LLN - ROOMS A	CEMS	Supply Chain	Innovation Management	Financial Engineering	Marketing Strategy for Connected Brands	Corporate Sustainable Management	International Finance (LLN)	CPME
Monday_8h30	0	0	Lavoisier-51	0	0	0	0	Agora-4
Monday_10h45	0	0	Barb-1	0	Barb-94	0	0	0
Monday_14h00	0	0	Agora-14	0	Doyen-32	0	0	Socrate-25
Monday_16h15	0	0	0	0	Doyen-32	0	0	Agora-4
Tuesday_8h30	BST-11	Agora-13	0	Lec1-70	0	Coub-11	Science-ETUS	0
Tuesday_10h45	BST-11	Agora-13	0	Lec1-70	0	Coub-11	Science-ETUS	Socrate-25
Tuesday_14h00	Sud-6	Agora-13	0	Barb-2	0	0	Sud-19	Socrate-21
Tuesday_16h15	0	0	0	0	0	Doyen-31	0	Doyen-32
Wednesday_8h30	0	0	Lavoisier-51	0	0	Doyen-31	0	Agora-4
Wednesday_10h45	0	0	Lavoisier-51	0	Barb-92	Doyen-31	0	Doyen-32
Wednesday_14h00	0	0	Agora-4	0	Doyen-32	Doyen-31	0	Barb-1
Wednesday_16h15	0	0	0	0	Doyen-32	0	0	0
Thursday_8h30	Lec1-51	Lec1-60	0	Coub-5	0	Mont-3	0	Agora-4
Thursday_10h45	Lec1-51	Lec1-60	0	Coub-5	0	Mont-3	More-53	Socrate-25
Thursday_14h00	Sud-6	Agora-12	0	Barb-3	0	Coub-11	Stud-13	Sud-7
Thursday_16h15	0	0	0	0	0	0	Stud-13	0
Friday_8h30	Agora-4	Lec1-61	Lec1-53	Coub-1	0	0	Stud-13	Mont-1
Friday_10h45	Agora-4	Lec1-61	Lec1-53	Coub-1	Sud-19	0	Stud-13	Barb-1
Friday_14h00	Agora-1	Agora-13	Agora-4	Barb-2	Doyen-31	0	Stud-13	Barb-0
Friday_16h15	0	0	0	0	Doyen-31	0	0	0

Trimester 2 LLN - LESSONS B	International Business (LLN)	Management humain (LLN)	European Business (LLN)	Corporate Finance (LLN)	Consumer Insights (LLN)	Option Sourcing and Procurement	Gest60 Tronc Commun
Monday_8h30	0	LLSMS2072	0	0	0	0	0
Monday_10h45	0	LLSMS2072	0	0	0	0	0
Monday_14h00	0	LLSMS2070	0	0	0	0	0
Monday_16h15	0	0	0	0	0	0	0
Tuesday_8h30	LLSMS2030	LLSMF2007	LLSMS2064	0	LLSMS2024	0	0
Tuesday_10h45	LLSMS2030	LLSMF2007	LLSMS2064	LLSMS2108	LLSMS2024	LLSMS2038	0
Tuesday_14h00	0	LLSMF2007	LLSMS2018	LLSMS2017	LLSMS2001	LLSMS2037	0
Tuesday_16h15	LLSMS2014	0	0	LLSMS2014	0	LLSMS2036	0
Wednesday_8h30	0	0	0	LLSMS2017	0	0	0
Wednesday_10h45	LLSMS2014	0	0	LLSMS2014	0	0	LLSMG2003
Wednesday_14h00	LLSMS2002	0	0	LLSMS2108	0	0	LLSMG2003
Wednesday_16h15	LLSMS2002	0	0	0	0	0	0
Thursday_8h30	0	0	LLSMS2060	0	LLSMS2021	LLSMS2037	LLSMG2002
Thursday_10h45	0	0	LLSMS2060	0	LLSMS2021	LLSMS2037	LLSMG2002
Thursday_14h00	0	0	LLSMS2018	0	LLSMS2024	LLSMS2036	0
Thursday_16h15	0	0	0	0	0	0	0
Friday_8h30	LLSMS2014	LLSMS2070	LLSMS2060	LLSMS2014	LLSMS2001	0	0
Friday_10h45	LLSMS2002	LLSMS2070	LLSMS2018	LLSMS2017	LLSMS2001	LLSMS2036	LLSMG2002
Friday_14h00	LLSMS2030	LLSMS2072	LLSMS2064	LLSMS2108	LLSMS2021	LLSMS2038	LLSMG2003
Friday_16h15	0	0	0	0	0	LLSMS2038	0

Trimester 2 LLN - ROOMS B	International Business (LLN)	Management humain (LLN)	European Business (LLN)	Corporate Finance (LLN)	Consumer Insights (LLN)	Option Sourcing and Procurement	Gest60 Tronc Commun
Monday_8h30	0	Agora-10	0	0	0	0	0
Monday_10h45	0	Agora-10	0	0	0	0	0
Monday_14h00	0	More-53	0	0	0	0	0
Monday_16h15	0	0	0	0	0	0	0
Tuesday_8h30	Agora-13	Mercator-28	More-77	0	Erasm-56	0	0
Tuesday_10h45	Agora-13	Mercator-28	More-77	Lecl-62	Erasm-56	Sud-1	0
Tuesday_14h00	0	Lecl-73	BST-11	Lecl-61	Lecl-62	Sud-1	0
Tuesday_16h15	Doyen-32	0	0	Doyen-32	0	P Curie-5	0
Wednesday_8h30	0	0	0	Lecl-61	0	0	0
Wednesday_10h45	Doyen-32	0	0	Doyen-32	0	0	Barb-12
Wednesday_14h00	Doyen-32	0	0	Agora-3	0	0	Barb-12
Wednesday_16h15	Doyen-32	0	0	0	0	0	0
Thursday_8h30	0	0	Mercator-12	0	Lecl-62	Mercator-1	Barb-21
Thursday_10h45	0	0	Mercator-12	0	Lecl-62	Mercator-1	Barb-21
Thursday_14h00	0	0	Erasm-56	0	Socrate-21	Agora-3	0
Thursday_16h15	0	0	0	0	0	0	0
Friday_8h30	Mont-1	More-53	Mercator-12	Mont-1	More-51	0	0
Friday_10h45	Sud-19	More-53	Mercator-12	Mont-1	More-51	Doyen-22	Socrate-40
Friday_14h00	Agora-13	Coub-12	Erasm-56	Lecl-62	Mont-2	Doyen-22	Agora-2
Friday_16h15	0	0	0	0	0	Doyen-22	0

