UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Automatic scheduling for master thesis defenses

Dissertation presented by
Ludovic FASTRÉ

for obtaining the Master's degree in
Computer Science

Supervisors
Pierre SCHAUS , François GLINEUR

Readers
Chantal PONCIN, Pierre REINBOLD

Academic year 2016-2017

Glossary

- TFE : Abbreviation representing "Travail de Fin d'Etudes" in French. It means end-of-study work in english and, in our case, it corresponds to the master thesis.
- Auditorium : The term "auditoire" is used in Belgium to designate a classroom used by students and teachers. So auditorium designate this room.
- Solver : Complex program used to solve a given problem.
- Advisor : Advisor is the same as supervisor for a master thesis.
- MIP : Abbreviation meaning Mixed Integer Programming, a class of optimization formulations.
- Juries : Readers or advisors attending the defense of a master thesis
- EPL : Abbreviation representing "Ecole Polytechnique de Louvain". This is the faculty where I study, but also the client of this project.
- Program commission (commission) : Structure of a faculty responsible for teaching and researching in a single disciplinary field.
- Secretary : Person in charge of the organisation of one or more commissions. They are involved in the process of making the schedule of the master thesis defenses.
- UCL : Abbreviation representing "Université catholique de Louvain". This is the University where I study.
- Python : Programming language used for the web tool and the backup system.
- Java : Programming language used for the solver.
- JSON : JavaScript Object Notation or JSON is a human readable file format that use text to transmit data objects.

Contents

Chapter 1

Introduction

Prior to 2017, the schedule of master thesis defenses was managed by the secretaries using only Excel tables and email exchanges. Each students could choose only one subject among those proposed by the commission of the faculty from which he/she depends. Actually, there are many commissions within the EPL faculty and each commission is headed by a single secretary.

Today, students can more easily select subjects from other faculties commissions and some subjects deal with themes involving several commissions. Scheduling would, therefore, be better managed jointly and not separately by each program commission. Coordinating between commissions with excel files sent by email only is not a suitable solution and can cause problems. With this kind of method, it is also difficult to have an overall view of all the master thesis (even those who do not belong to the student's commission). Therefore, it would be very useful to have an automated tool to prepare the defenses thesis schedule.

We are facing two main problems: assigning a commission to each TFE (and therefore designate a secretary to take care of) and giving, optimally, an hour plus one auditorium for each defense. The results related to these two problems and generated through an intelligent algorithm are only proposals. Therefore, the aim of this master thesis is also to create a tool allowing the secretaries to modify the results. You will see it later in the report, but that is also why there are two independant programs to be developed : the solver and the web tool. And of course they must communicate with each other.

My main motivation for choosing this master thesis is that my work will be useful in the short term and will help a lot of people. The result will be directly effective as the program will be used already for the sheduling of the defense of master thesis in June 2017. Its feedback would also be really fast since it is for the EPL.

This report will show the steps followed to meet the objectives of this master thesis (ch. 2). It will detail the information needed before entering the development phase. Then the way the time was managed as well as how I divided the work to accomplish (ch. 3).
It will also show the global architecture of the project, and how the different modules communicate with each other (ch. 4). I will explain in more details the two problems encountered and how I managed to solve them. The first problem (ch. 5), commissions distribution, is clearly trivial because it doesn't require any artificial intelligence. But for the second problem (ch. 6), the scheduling in itself, I will introduce the model I established with the different variables, constraints and the objective function. Then I will show the database model used for the website (ch. 7), how the website works (ch. 8) and how the users can use it concurrently (ch. 9). I will also explain how to deploy the application and talk about the backup system (ch. 10).
To conclude, I will present the tests and how they were performed (ch. 11) as well as all im-

provements that can be made (ch. 12) and the opinion of the users on the application (ch. 13). For each technical part, I will develop the technologies used and why they were chosen.

Chapter 2

Objectives

The aim of this thesis is to create a tool allowing secretaries to create, coordinate and publish the schedules of the TFE defenses. The schedule is composed of several small sessions of 2h30 each. Each session possesses a maximum of 3 master thesis. It is necessary to take into account the composition of juries, the availability of members. Moreover, it is interesting to minimize the number of sessions to which each member of the jury must participate.

In order to achieve this result, the proposed solution must fulfil several objectives :

- Collect constraints
- Propose a distribution of the TFE's to the different commissions
- Propose a schedule automatically
- Validate the schedule or make adaptations
- Recalculate the schedule partially in case of adaptations
- Distribute the schedule to students and jury members
- Publish the schedule (posters, broadcast on screen, ...)

In order to solve the problem, I had to create a model and use a solver to get a first solution. The goal is of course to reach the optimality, I will describe all that part in the Mixed Integer Programming (MIP) solver section (ch. 6).
After that, in order to have a solver easily usable, a good and clean interface is needed. This is the web tool part. With this tool, we will be able to see the results of the solver in a nice way, but we will also be able to modify the data related to each master thesis.
The first three points of the objectives concern the solver part creating the schedule. And the 4 last ones concern the web tool part.

Chapter 3

Organisation of the master thesis

This master thesis subject is not a research thesis. It is about a scheduling tool that the client (UCL) will use in order to solve a complex problem and save time while organizing this schedule. Therefore, organisation and preparation of the working planning are important issues to be also taken into account.
Since the UCL is the client, it was fairly easy for me to have regular feedback at every stage of the implementation of the scheduling tool.
The first thing to do was to contact the client (Ms Chantal Poncin) in order to gather information on the problem ; i.e why is this application needed and used, what are its objectives, its global specifications, the deadline, etc...
Then, I had to meet the future users of the application (Ms Vanessa Maons, Ms Chantal Poncin) in order to collect specifications and constraints in the model to be used.
After these two meetings, the next thing to do was to present the problem to the supervisor (Prof. Pierre Schaus) and to propose a first model. During that meeting it was decided to define four deadlines using the Agile method (cf. 3.1):

1. End of October :
    - elaborate an input/output format in JSON
    - an output checker
    - a solver for the TFE distribution
    - generate unit tests
2. Mid November :
    - modeling the scheduling problem
    - building a program that solves it
3. End of December :
    - Elaborate a schema for the database
4. Beginning of February :
    - first prototype of the online version
    - presentation to the client

After that point, from early February to early April I used to meet every 2-3 weeks, following the Agile method, with Prof. Pierre Schaus (supervisor), Prof. François Glineur (supervisor), Ms Chantal Poncin (client), Ms Vanessa Maons (user) and Mr Ludovic Taffin (sysadmin[1]). This

[1] permanant system administrator to the UCL

latter will be in charge to maintain the application after September 2017.

When the development phase was completed, it was necessary to organize a testing phase. At that time I also started writing my master thesis report. For the organization of the tests, we divided them into several phases corresponding to the different routine steps used by the secretaries for the organization of the master thesis defenses schedule:

1. give access to people attending meetings
2. encode data in the application
3. encode availabilities in the application
4. make the schedule

This thesting phase was very important for the proper functioning of the program and is detailed in section 11.2.

Eventually, the production phase began on May 8th and the final schedule created with the help of the application was released on May 24th with the last changes.

3.1 Agile method

First of all, an Agile method, like SRUM [9], is a set of principles for software development. Compared to other methods, this method focuses on the development with dynamic, non-deterministic and non-linear characteristics. Therefore, the Agile method was chosen for the organization of this project as this project would be confronted with many changes during the development, this development took place over a short period and would always ask for the client interaction. So how was the Agile method applied concretely for this project ?

As you may have seen, the project has been always divided into several two-week periods called "sprints". A sprint always starts and ends with a meeting. At the opening meeting, the team would define which features of the program needed to be implemented and in what way. These sprints would allow the team to stay up-to-date on the progress of the project and to express their point of view. With this, the application would evolve and could fulfill the wishes of the customer. However, throughout a complete period of a sprint the objectives remain the same. Improvements and/or changes were only decided at the end of each sprint. This approach guaranteed the structure in the project and its delivery on time.

Thanks to this Agile method, the application was delivered on time while meeting the needs of the customer.

Chapter 4

Global architecture

To better understand the architecture of the program, it is necessary to know that the latter is separated into two large modules. The first module is the part in which the user will interact. This is obviously the web tool that includes a server and a database to store all information on TFE's. The second module will concern the solver. The solver, on the basis of the input, will be responsible for sending back a proposal for a schedule of master thesis defenses.

The program is separated in two parts for two reasons. The first reason is that the solver can be used independently. Indeed, if you wish, you can run the solver by inputing a Json file with all the information needed by the module (this part is explained in detail in the next section). As a result, reusability is guaranteed. The second reason is that Gurobi [3] (the library used by the solver), at the time of writing the solver, did not exist in Python. It is only very recently (early 2017) that Gurobi could be supported in Python. Therefore this module has been developped in Java.

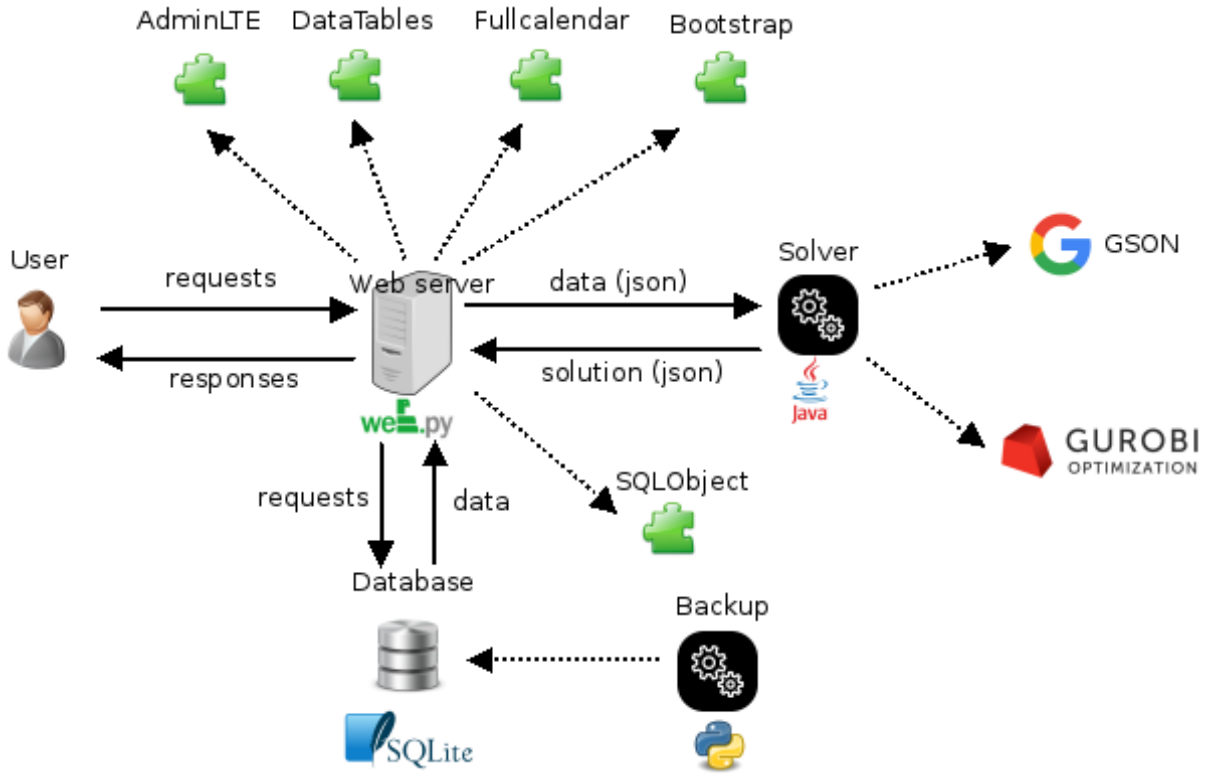


Figure 4.1: Global architecture

As you can see on figure 4.1, everything begins with the user requests. This user will perform queries (read the schedule, create a TFE, ...) that will be processed by the server.

This server has been realized using the framework Webpy [10] and uses several plugins such as AdminLTE [1] for the global graphical interface, Datatables [7] to display the data, Fullcalendar [6] for the display of the schedule or Bootstrap [11] to have a responsive interface[1].

The server accesses the data via a database in SQLite [4] using the SQLObject[2] [2] ORM (Object Relational Manager). The database will also be saved at regular intervals by a taylored backup program written in Python.

If the user has made a request to run the solver, the server will call the solver by sending to it a JSON file containing all the data needed. The solver, as soon as it has finished running, will give to the server the solution it has found. The server will then treat the information, save it in the database and display it to the user. The solver uses the Google Gson [5] library to parse the JSON file because there is no JSON parser in java (only third parties libraries). And of course, the solver will use Gurobi [3] which will allow to find an optimal solution to the problem.

All the libraries used in this project are free to use for academic purposes or for open source programs. The two modules (web tool and solver) are open source and can be found at the addresses : `https://github.com/Siceron/webTFEScheduler` and `https://github.com/Siceron/TFEScheduler` under the MIT license.

4.1 Communication between the web tool and the solver

Since the solver is coded in Java, the source code can't be integrated directly into the website. But this is not a problem. Actually, we can use the subprocess library in Python that allows us to run a sub process shuch as an executable jar with Java and this is what is wanted as the solver is compiled in a jar file. This process can be called like that :

```
Popen(["java", "-jar", "scheduler/TFEScheduler.jar", "input.JSON",
 x.time], stdout=PIPE, stderr=STDOUT)
```

As you can see, the argument "input.JSON" for the java process was used. This is because the web server is passing a JSON formatted file to the solver. This JSON file contains all the data needed for the solver to work (cf. annex 15.1) :

- reserveDay : The reserve day is the day where one should avoid to assign a TFE. It is an integer representing the index of the day (0 being the first day).
- sessionDay : This is the number of days. It is important to know this information in order to compute the number of sessions : 4 * sessionDay * sessionRooms (4 because there are 4 sessions in a day in the current state of the application).
- sessionRooms : It is the number of auditoriums available for the schedule. It is the variable used to compute the number of sessions.
- advisors :

[1] All of this will be explained in detail in the web tool section
[2] Will also be explained in detail in the web tool section

– email : This is the email address of the advisor (unique). It is used for identification.
– disponibilities : This is an array of the availabilities of the advisor used for the constraints.

- readers :
  - email : This is the email address of the reader (unique). It is used for identification.
  - disponibilities : This is an array of the availabilities of the reader used for the constraints.
- tfes :
  - code : This is the identification code of the TFE (unique).
  - students :
    - email : This is the email address of the student (unique). It is used for identification.
    - faculty : This is the faculty the student depends of. It is used to determine the Program Commission of the TFE.
  - advisors : An array containing the advisor's email address of the TFE.
  - readers : An array containing the reader's email address of the TFE.
- fixed :
  - code : The code of the TFE that need to be fixed.
  - session : The session concerned.
- banned :
  - code : The code of the TFE that need to be banned. By banned, it means that the tfe cannot be assigned to a specific session (but the banned functionality is not used in the web tool).
  - session : The session concerned.

With all this information, the solver is able to perform the optimization of the schedule within a time limit.
After that, the solver will return a report (text file) with the different conflicts and statistics such as the ratio of TFE assigned. It will also return the TFE's assigned in a new JSON file : output.JSON (cf. annex 15.2). This JSON file contains each code of the TFE and the session associated to it. A session equals to -1 means that the TFE is not assigned.

Giving a JSON file as input to the solver implies that it can be used independently of the web tool. Furthermore, since the JSON format is human readable, the input file can be easily modified by the user.

Chapter 5

Commissions distribution

The Program Commission distribution problem is very simple. Every TFE must be assigned to a Program Commission (commission). It will tell which secretary is in charge of this TFE.

At first, the idea was to first take into account the faculty of the supervisors and then the faculty of the students. It was sufficient to define a weight for each actor of the TFE and take the commission of the majority.

But in the end, the solution adopted was even simpler. The student commission was all that mattered and therefore the algorithm is very simple. Basically, it will look at every master thesis and take the commission of the majority of the students. For example, if two students are in SINF, it will attribute the SINF commission to the TFE. But, if the two students are in different commissions, it will attribute the "TBD" (to be determined) commission to the TFE. Only the secretaries have the last word, it is their responsibility to modify the commission of a TFE or define to which commission depend a TFE with the "TBD" argument. The program only suggests a solution.

The commissions distribution is part of the solver module and therefore can be used independently of the web tool. Just give it a JSON file as input and it will return the solution as JSON.

Chapter 6

Scheduling : Model and MIP Solver

6.1 Technology and tools selection

At first, since there were many constraints to be taken into account, constraint programming seemed to be the best solution for this problem. Enumerating all the constraints led to realize that the model was linear. And therefore, linear solver was used (linear programming). And since only integers for the variables were used (actually binary variables, since a TFE set to a session is equal to 1), mixed integer programming seemed to be the best solution.
In order to solve this problem on a machine, I decided to use the Gurobi [3] solver. It seemed unnecessary to reinvent the wheel while tested and approved solutions existed. The Gurobi optimizer is an optimization solver for linear programming, quadratic programming, quadratically constrained programming, mixed integer linear programming (the one who we are interested in), mixed-integer quadratic programming and mixed-integer quadratically constrained programming.
At that time, the only languages that supported the optimizer were C++, Java and .NET. Since I'm more comfortable with Java and there's a lot of documentation with this language, I chose to use it. If you would like to know how gurobi's search operate in detail, see the annex 15.3.

6.2 What is Mixed Integer Programming ?

A linear mixed integer program is an optimization problem which contains only integer variables. The constraints of the model are linear equalities or inequalities and the objective function is also a linear function to be minimized or maximized. Additional information can be found in the Mixed Integer Programming book of Laurence Wolsey [12].

6.3 Choice of model

In order to achieve optimality in the most appropriate and fastest way, two different models with the same variables were created.

The first model assumes that each TFE should be assigned to a session. This model reduces the complexity of the problem. However this can lead to an infeasible model if no solution meets the constraints.
The second one allows not to assign some TFE's while trying to reduce their numbers. The consequence would increase the number of possible solutions and therefore the complexity of the problem.

So, at first, the solver will try to solve the problem using the first model in which each TFE should be assigned. If the model is infeasible, which should take a matter of seconds to detect,

the solver will try to solve the problem using the second model.

6.4 Preprocessing

As mentioned previously, the second model is rather complex and its use should be avoided. In order to comply with this challenge, preprocessing some TFE's to make the model feasible was necessary. These TFE's will be removed from the domain of the problem and will be consired as not assigned.
There are two cases of TFE to preprocess. The first is fairly obvious: if the combination of the jury availabilities leaves no possibility to assign a TFE to a session, then this TFE must be removed from the domain of the problem.
The second is a little bit more tricky : if the availabilities of a jury member is not sufficient to the number of TFE's allocated to him/her, then all those TFE's should be removed from the problem domain. For example, Professor X is assigned to 4 TFE's. But he is available to attend only one session. Since there can be maximum 3 TFE's per session, then the problem is infeasible (actually, this case happened in the production phase). But then why remove the 4 TFE's while removing only one would suffice? It was decided, in this particular case, that rather than letting the program choose arbitrarily which TFE to withdraw, we would leave that choice to the secretaries. With this, the first model should be able to find a solution because the particular cases are discarded. But there could be other causes of infeasibility (it never happened in practice), that is why the second model is still present for those cases.

6.5 Models

Now we will show you more formally the two models mentioned above. The first one saying that all TFE's should be assigned. And the second one, giving more tolerance on this last point.

6.5.1 Model 1

Variables

- $M_{i,t} \in \{0, 1\}$: Master thesis i for session t (assigned or not)
- $P_{j,t} \in \{0, 1\}$: Jury j for session t (assigned or not)
- $S_t \in \{0, 1\}$: Session t (used or not)

Data

- M_i : master thesis i
- P_j : jury j
- $D_{P_{j,t}}$: availability of jury j at session t
- n_t : number of sessions
- n_p : number of jury members
- n_m : number of master thesis
- n_r : number of rooms
- m_{p_j} : number of master thesis for P_j

- *reserve* : the set of sessions we should avoid to use
- γ : weight for a master thesis assigned in the reserve
- θ : weight for the number of sessions used

Objective

To minimize the number of sessions assigned to a jury member (weighted by the number of master thesis he has to attend : it is more important to try to reduce the number of sessions for a jury member with 9 TFE's which he/she has to attend than for one with 2 TFE's), the number of sessions used (weighted by θ) and the number of TFE's in the reserve day (weighted by γ) (6.1). Since a session is composed of maximum 3 master thesis, $m_{p_j}/3$ is the minimum number of sessions assigned to a jury member.

Constraints

- There are more jury members than master thesis for a session t (6.2)
- A master thesis must be assigned to a session (6.3)
- Maximum 3 master thesis per session (6.4)
- The jury member j must be available at the session t (6.5)
- The jury member j cannot have parallel sessions (6.6)
- The jury member j assigned to the master thesis i must have the same session than i (6.7)
- A session S_t is used if a master thesis is assigned at the session t (6.8)

Model

$$Minimize \sum_{j=0}^{n_p} \left[\left(\sum_{t=0}^{n_t} [P_{j,t}] - \frac{m_{p_j}}{3} \right) * m_{p_j} \right] + \sum_{t=0}^{n_t} [\theta * S_t] + \sum_{i=0}^{n_m} \left[\sum_{t=0}^{n_t} [\gamma * M_{i,t} \ (if \ t \in reserve)] \right] \tag{6.1}$$

$$\sum_{i=0}^{n_m} M_{i,t} \leq \sum_{j=0}^{n_p} P_{j,t} \quad \text{for} \quad t \in [0, n_t[\tag{6.2}$$

$$\sum_{t=0}^{n_t} M_{i,t} = 1 \quad \text{for} \quad i \in [0, n_m[\tag{6.3}$$

$$\sum_{i=0}^{n_m} M_{i,t} \leq 3 \quad \text{for} \quad t \in [0, n_t[\tag{6.4}$$

$$P_{j,t} \leq D_{P_{j,t}} \quad \text{for} \quad j \in [0, n_p], t \in [0, n_t[\tag{6.5}$$

$$\sum_{l=0}^{n_r} P_{j,k+l*(\frac{n_t}{n_r})} \leq 1 \quad \text{for} \quad j \in [0, n_p], k \in \left[0, \frac{n_t}{n_r}\right[\tag{6.6}$$

$$M_{i,t} \leq P_{j,t} \quad \text{for} \quad i \in [0, n_m[, t \in [0, n_t[, j \in M_i \tag{6.7}$$

$$M_{i,t} \leq S_t \quad \text{for} \quad i \in [0, n_m[, t \in [0, n_t[\tag{6.8}$$

6.5.2 Model 2

Variables

- $M_{i,t} \in \{0,1\}$: Master thesis i for session t (assigned or not)
- $P_{j,t} \in \{0,1\}$: Jury j for session t (assigned or not)
- $S_t \in \{0,1\}$: Session t (used or not)

Data

- M_i : master thesis i
- P_j : jury j
- $D_{P_{j,t}}$: availability of jury j at session t
- n_t : number of sessions
- n_p : number of jury members
- n_m : number of master thesis
- n_r : number of rooms
- m_{p_j} : number of master thesis for P_j
- η : weight for a master thesis assigned
- $reserve$: the set of sessions we should avoid to use
- γ : weight for a master thesis assigned in the reserve
- θ : weight for the number of sessions used

Objective

To minimize the number of sessions assigned to a jury member (weighted by the number of master thesis he has to attend), the number of master thesis not assigned (weighted by η), the number of sessions used (weighted by θ) and the number of TFE's in the reserve day (weighted by γ) (6.9). Since a session is composed of maximum 3 master thesis, $m_{p_j}/3$ is the minimum number of sessions assigned to a jury member.

Constraints

- There are more jury members than master thesis for a session t (6.10)
- One master thesis can be assigned to only one session (6.11)
- Maximum 3 master thesis per session (6.12)
- The jury member j must be available at the session t (6.13)
- The jury member j cannot have parallel sessions (6.14)
- The jury member j assigned to the master thesis i must have the same session than i (6.15)
- A session S_t is used if a master thesis is assigned at the session t (6.16)

Model

$$Minimize \sum_{j=0}^{n_p} \left[\left(\sum_{t=0}^{n_t} [P_{j,t}] - \frac{m_{p_j}}{3} \right) * m_{p_j} \right] + \sum_{i=0}^{n_m} \left[\sum_{t=0}^{n_t} [-\eta * M_{i,t}] \right] + \sum_{t=0}^{n_t} [\theta * S_t] + \sum_{i=0}^{n_m} \left[\sum_{t=0}^{n_t} [\gamma * M_{i,t} \ (if \ t \in reserve)] \right] \quad (6.9)$$

$$\sum_{i=0}^{n_m} M_{i,t} \leq \sum_{j=0}^{n_p} P_{j,t} \quad \text{for} \quad t \in [0, n_t] \quad (6.10)$$

$$\sum_{t=0}^{n_t} M_{i,t} \leq 1 \quad \text{for} \quad i \in [0, n_m] \quad (6.11)$$

$$\sum_{i=0}^{n_m} M_{i,t} \leq 3 \quad \text{for} \quad t \in [0, n_t] \quad (6.12)$$

$$P_{j,t} \leq D_{P_{j,t}} \quad \text{for} \quad j \in [0, n_p], t \in [0, n_t] \quad (6.13)$$

$$\sum_{l=0}^{n_r} P_{j,k+l*(\frac{n_t}{n_r})} \leq 1 \quad \text{for} \quad j \in [0, n_p], k \in \left[0, \frac{n_t}{n_r}\right] \quad (6.14)$$

$$M_{i,t} \leq P_{j,t} \quad \text{for} \quad i \in [0, n_m], t \in [0, n_t], j \in M_i \quad (6.15)$$

$$M_{i,t} \leq S_t \quad \text{for} \quad i \in [0, n_m], t \in [0, n_t] \quad (6.16)$$

6.6 Performances

Before showing the performances, it is useful to explain some concepts. Performances will be based on 3 measures: time, goal and gap.

The objective will show us how close we are to an optimal result. It is based on what has been said above with the first model. We wanted all the TFE's to be assigned, this is why we used the first model. Both models have different objectives, so it is complicated to compare them objectively.

Since we try to minimize the goal, it is important to have an upper bound when searching (branch and bound algorithm). In this case, the solver knows that it will never have to accept a solution of value higher than this upper bound. But, at any time during the search, we also have a lower bound (also called best bound). This lower bound is obtained by taking the minimum of the optimal objective values of all of the current leaf nodes. The difference between these two values is called the gap. If this gap is zero, then we have reached optimality.

Now that all the concepts have been defined, we will be able to observe the different performances depending on the size of the problem.

6.6.1 Performances with 149 TFE's and 278 jury members

The data below is test data and the availabilities of the jury members were randomly generated on the basis of the availabilities of the previous years.
These tests were performed on a mac computer (CPU: 2.4GHz i5, memory: 4GB) during 30 minutes.

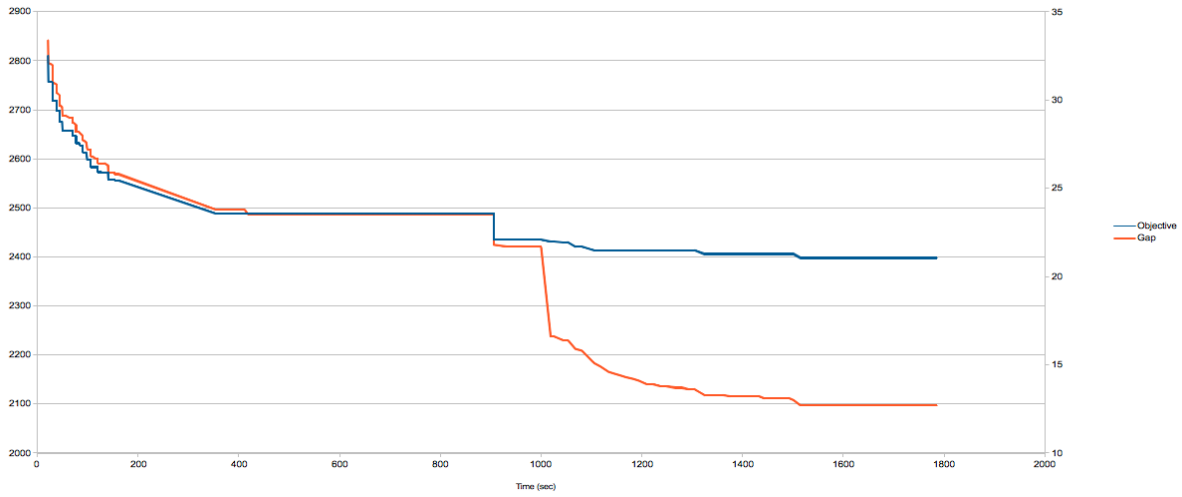


Figure 6.1: Performances with 149 TFE's, 278 juries and 5 auditoriums

First we will calculate the performance with 5 auditoriums available at every time slot. We chose to use 5 because it is the minimum required to have all TFE's assigned: $149/(12*3) = 4.14$ (149 TFE's, 12 sessions per auditorium and 3 TFE's per sessions).

As you can see in the figure 6.1, as from the start of the query and during 353 seconds the solver would find increasingly better solutions. At the 906 second the solver will find a better solution, but from this moment the gap will decrease faster than the objective, which means we are probably close to an optimal solution. The execution ends with an objective of 2398 and a gap of 12.7%.

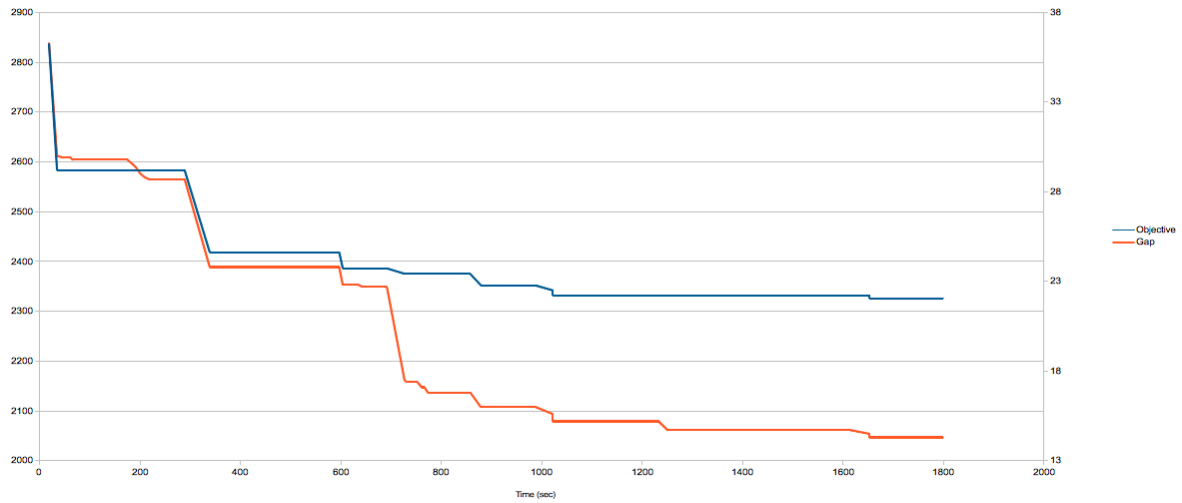


Figure 6.2: Performances with 149 TFE's, 278 juries and 8 auditoriums

Then we calculated the performance with 8 rooms available. We chose this number because it makes it possible to avoid assigning TFE's in the reserve day and leaves more freedom to the solver.

As you can see in the figure 6.2, as soon as the solver finds a better solution the objective decreases drastically. But from 604 seconds, the gap decreases faster than the objective. However, it will continue to find better solutions, but in a less significant way which suggests that we are close to an optimal solution. The execution ends with an objective of 2326 and a gap of 14,3%.

In conclusion, in the short term, the first case will find solutions more quickly. On the other hand, the second case will find more optimal solutions but in the long run. This is probably related to the fact that the solver has greater freedom in the second case: the search space is bigger, but allows to better satisfy the objective function.

6.6.2 Performances with 256 TFE's and 394 jury members

These data are test data and the availabilities of the jury members were randomly generated on the basis of the availabilities of the previous years.
These tests were performed on a mac computer (CPU: 2.4GHz i5, memory: 4GB) over 30 minutes.

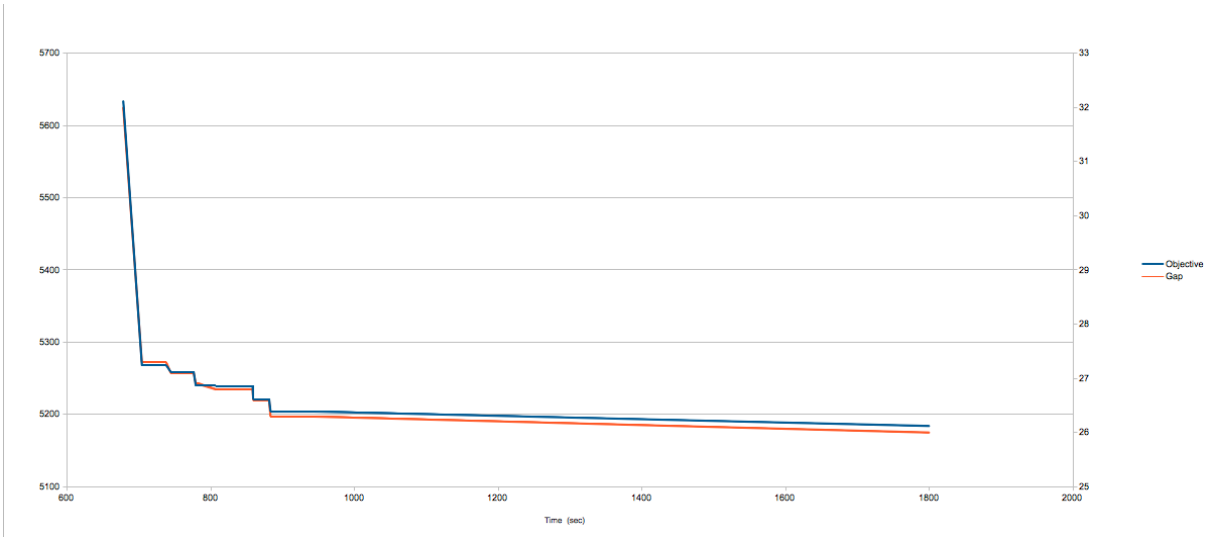


Figure 6.3: Performances with 256 TFE's, 394 juries and 8 auditoriums

We will calculate the performance with 8 auditoriums available. We chose to use 8 because it is the minimum if we want all the TFE's to be assigned: $256/(12 * 3) = 7.11$ (256 TFE's, 12 sessions per auditorium and 3 TFE's per sessions).

This problem becomes very complicated to solve because we have to assign 256 TFE's and 394 jury members who may have different availabilities to different sessions.

As you can see in the figure 6.3, the solver took more time to find a solution than in the previous tests. The objective is followed very closely by the gap, which means that according to our latest observations we are perhaps not yet close to the optimal solution. Obviously the solver needed more time, but it already found solutions when it only turned 30 minutes.

After this, we tried to execute the solver with the same data but using 12 auditoriums instead of 8. The solver failed to find a solution on time. This reinforces the fact that the solver is more

likely to find a solution if the number of auditoriums available is at its minimum (smaller search space). It will probably find better solutions, but 30 minutes are not enough in this case.

6.6.3 Performances of the final result

This data is real data used for the master thesis defenses of June 2017.
This execution was performed on a linux server (CPU: 2.4GHz quad, memory: 4GB) over 24 hours.

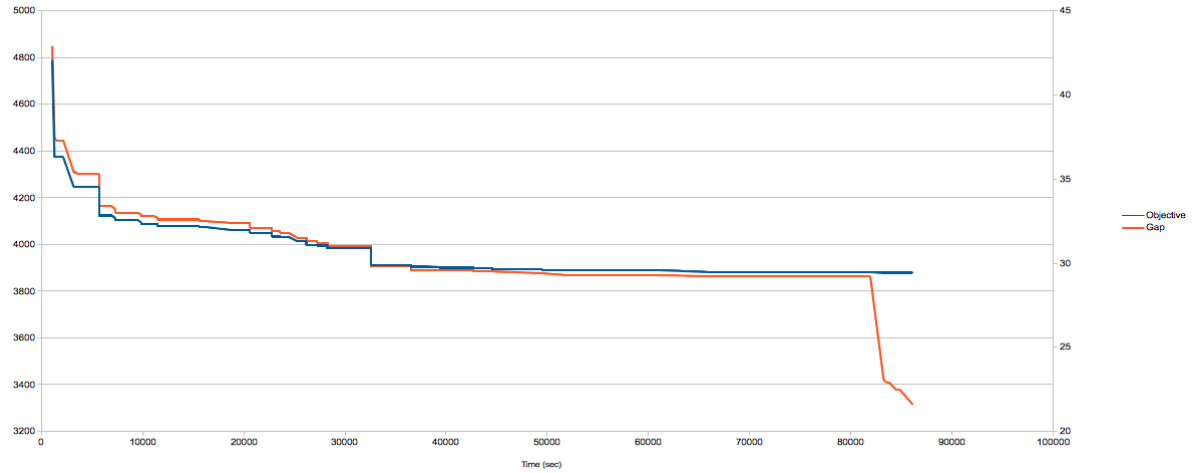


Figure 6.4: Performances of the final result

And as you can see in the figure 6.4 that the solver takes time to find a first solution and to converge towards the best objective. This is mainly because the problem contains a lot of variables (256 TFE's and 394 juries over 12 auditoriums) and has a lot of things to optimize (see objective 6.5.1).

The solver stopped with a gap of 21.6%. That sounds a lot, but for a problem of this size, it's acceptable. And as you can see, as in the tests, the gap decreases faster than the objective at a moment (at 83253 seconds). It means that we may be at the best objective. Actually, we could have stopped at 32581 seconds because the solver gave us a very satisfactory answer.

Chapter 7

Database model

In order to understand what will follow, I have to talk about the database. It is the behind the scene of the website, where all the permanent data is stored.

Here is a simple table relation diagram representing all the tables of the database :

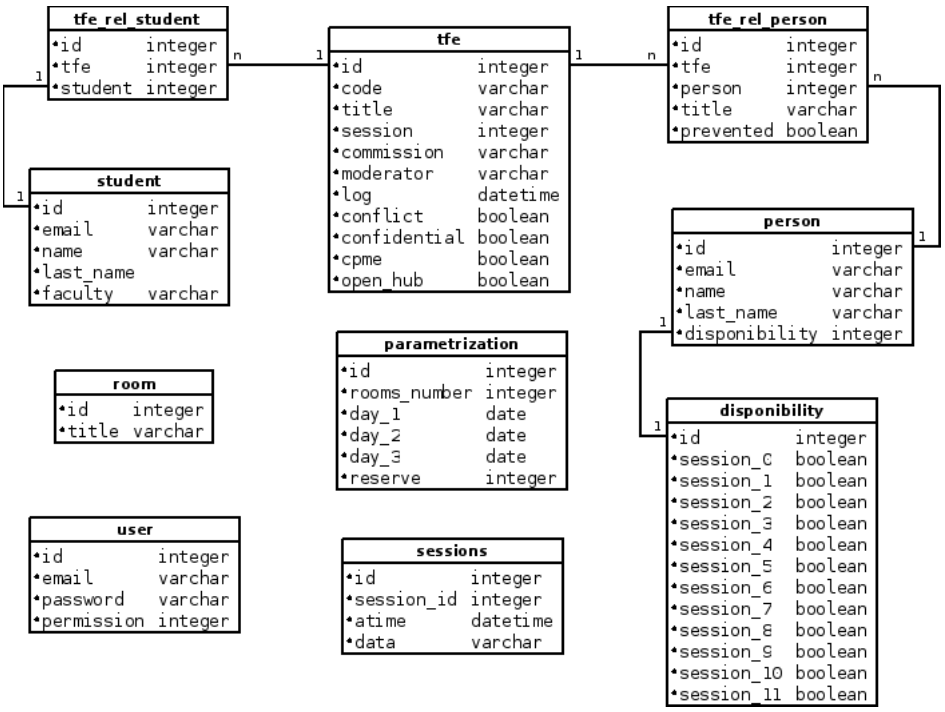


Figure 7.1: Table relation diagram of the database

As you can see in the figure 7.1, there are 10 tables :

- user : the credentials of the user and his/her permissions (0 : normal user, 1 : admin). In the future it may have more types of permission, that's why I used an integer instead of a boolean. And password stored is hashed with the md5 algorithm (cryptographic hashing).
- session : this table is used by the Webpy framework in order to store the session informations.
- parameterization : it concerns the parameterization of the solver. We need to enter this data before performing the automatic scheduling. At this stage of the application, there are 3 days possible for the scheduling. Maybe in the future it would be better to remove the limit of days. The variables are self explanatory, except for the reserve : this integer

represents the reserve day. Also, in the future, it may be better to give the possibility to any day to be a reserve day.

- tfe : this is the most important table. It contains the basic informations on the master thesis. Its code is unique and is used everywhere in the application to distinguish the different TFE's. The session is of course the number of the session. This number is between 0 and 3(number of days)*4(sessions per day)*rooms_number. Then we have the commission given by the solver or "TDB" (to be determined) if the solver didn't find a match. The log is actually the timestamp of when the TFE was last changed. Conflict is a boolean representing the fact that a TFE fixed has some conflicts the session it has been assigned to. The rest of the variables are simply stored data.

- student : all the informations concerning the student. His/her email address, firstname (name), lastname and faculty (used for the commission solver). The email address is used as an identification of the student since it's unique.

- tfe_rel_student : it is a relation between the student and his/her TFE. As you can see, the student has only one TFE link.

- person : all the information the specific person (advisor or reader, basically a jury member). This is the same information as the one provided for the student except that the person is not assigned to a faculty (not needed) and that he/she has availabilities (disponibility) to be taken into account. This availabilities are used for the constraints of the solver.

- disponibility : These are the different availabilities of a person. As you can see there are 12 variables. This is due to the fact that we have 3 days and 4 sessions in a day (so 3*4). This number needs to be more general and it has been added to the improvements to bring to the application (ch 12).

- tfe_rel_person : is the relation between a person and a TFE. A person can be linked to multiple TFE's (for example a professor can be reader of a TFE and advisor to an other one). There is a title variable (advisor or reader) in order to know the role of the person for the TFE. It could be changed to a boolean in order to spare space. And finally there is the prevented variable saying that we can assign the TFE to a session even if the person of the relation is in a conflict situation.

- room : this table simply stores the name of the auditoriums.

Chapter 8

Web tool

The sysadmins suggested me to use Webpy [10] as framework, the tool that they use. The maintenance/development of the application will also be easier.
The use of Webpy is really easy, you import a package called web and then you can define all the urls in one place. You can easily create functions for each POST[1] or GET[2] request. And after that, you can call variables from your Python code in your html[3] template using the "$" symbol. The framework is also compatible with SQLObject. You just have to import your models into your web file. All this makes it simple to use, to understand and to maintain.

8.1 Technology and tools used

In web development and elsewhere, it is sometimes better to use existing libraries / frameworks rather than reinventing the wheel. This will not only save you time needed for another feature, but these solutions are as well tested and robust as they are used by many people. This is why some plugins have been used during the development of the website.

The first one is the Datables [7] plugin used to show the data from the database. The plugin is complete, flexible and well documented. This plugin only serves to show the data in an elegant way. So it's a plus for the user, but not something essential.

The second one is the Fullcalendar [6] plugin used to have a good vizualization tool for the schedule. It is a good start for a first production, but this plugin lacks in flexibility. In the future, it would be better to code an interface adapted to the problem. But the use of this plugin made it possible to advance faster than expected on the project and allowed to focus more on more important areas.

The third one is Bootstrap [11]. This one is well known and used in the web development domain. It is used in order to have a simple, nice and responsive interface. By responsive, it means that the website will always looks good, whether on small or large screens.

The fourth on is a template called AdminLTE [1]. It is basically .html and .css files that you can be customized. It is the base of the graphical interface and it uses Bootstrap, so it is responsive.

[1]POST is a request method supported by the HTTP protocol. POST send info to the server.
[2]GET is a request method supported by the HTTP protocol. GET receive info from the server.
[3]HyperText Markup Language : data format used to represent web pages

The last one is the SQLObject [2] ORM (Object-Relational Mapping) used to make easy calls to the database. This one was suggested by the sysadmins as it is the one used with Webpy. This is pretty handy because it is coded in Python as Webpy but its documentation about this ORM is rather bad and looks more like a tutorial. Some functions are not even listed and you have to dive into the source code to find your answers. This ORM works with SQLite [4] and this is a good choice because it needs no configuration, it is serverless, all the data are stored in a .db file (easy for backups), the database file can be used on machines with different architectures and SQLite is opensource. Moreover, since the database will be relatively small, SQLite seems to be a good choice.

8.2 Pages

8.2.1 Login page

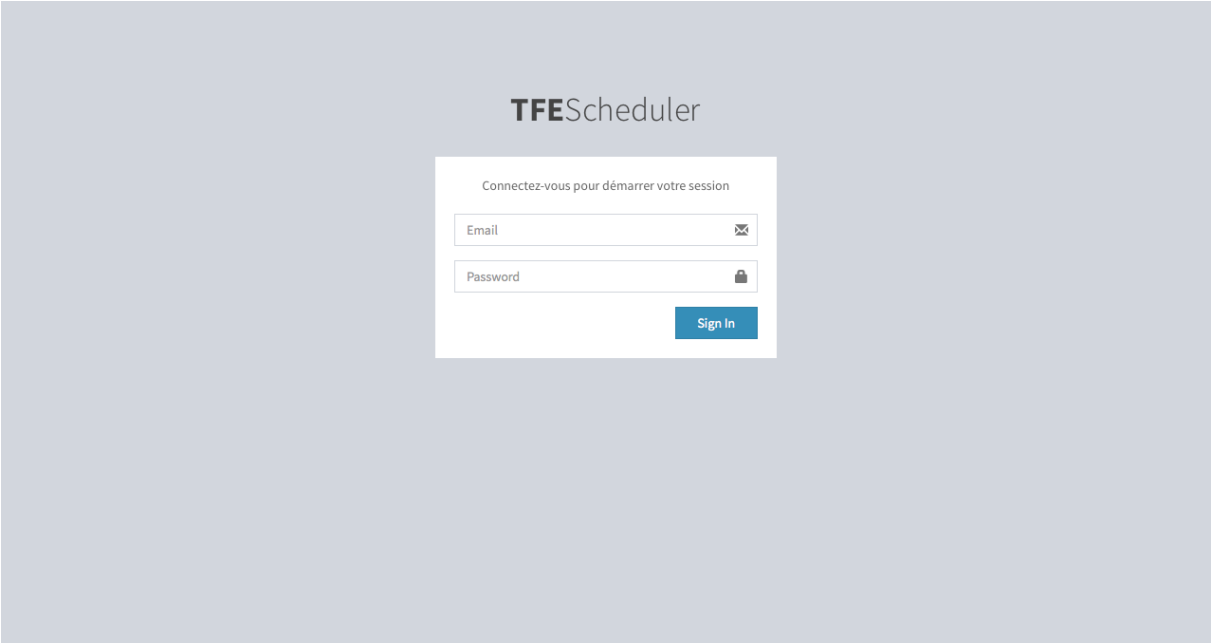


Figure 8.1: The login page

This page (figure 8.1) is the first that appears once you enter the url of the website (currently http://130.104.78.219:8080). This is the login page where email addresses and passwords are entered once an account has been created for the specific user. A login page is primordial if you want to have only authorized users.

The use of this page is pretty simple : you enter your email address, your password and then you hit the "sign in" button or "enter". Should your credentials not be correct, the server will send you a message mentioning that the information input is not correct. On the other hand, if your information is correct, it will redirect you to the dashboard page. Doing so, the server will create a session object containing your identification (email address) and the privileges you have. Later, we will see that the server stores also a timestamp for the concurrent utilisation of the data.

8.2.2 Index page

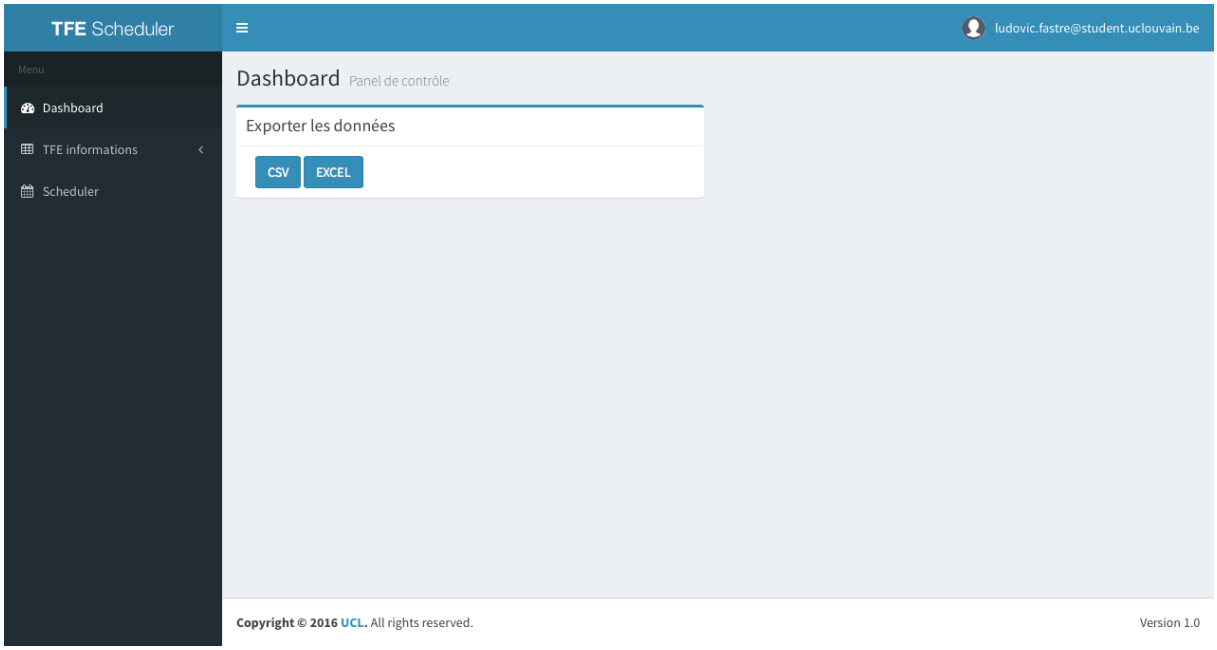


Figure 8.2: The common index a.k.a. the dashboard

This page (figure 8.2) is displayed when the correct credentials are input and when permissions are restricted. This is the "tool" page where some functionalities will help continuing to work. There is not much choice here, but secretaries don't have full access notably to the functionalities that will be developed in the next section.

Only extraction of data on a table (excel or csv) is allowed. This functionality was chosen as sometimes only a global view of the data is needed. Should the application not meet the expectations of the user, the previous way to schedule can still be used. This extraction data on a table will also, at the end of the day, give a suggestion for the future the publication of the useful timetable for the juries and for the students.

On this page, as well as on the following ones, the menu is displayed on the left of the screen. On this menu, you can access the dashboard (the current page), the TFE information (TFE, students, jury and auditoriums) and the scheduler. These pages will be shown on the next sections. The menu can be minimized by clicking on the three bars above the Dashboard title (it is useful if you have a small screen and you want to better see the content). On the top right of the screen, the email address us for connecting is displayed. Clicking on it, will enable the user to disconnect from the session and to be redirected to the login page.

8.2.3 Admin index page

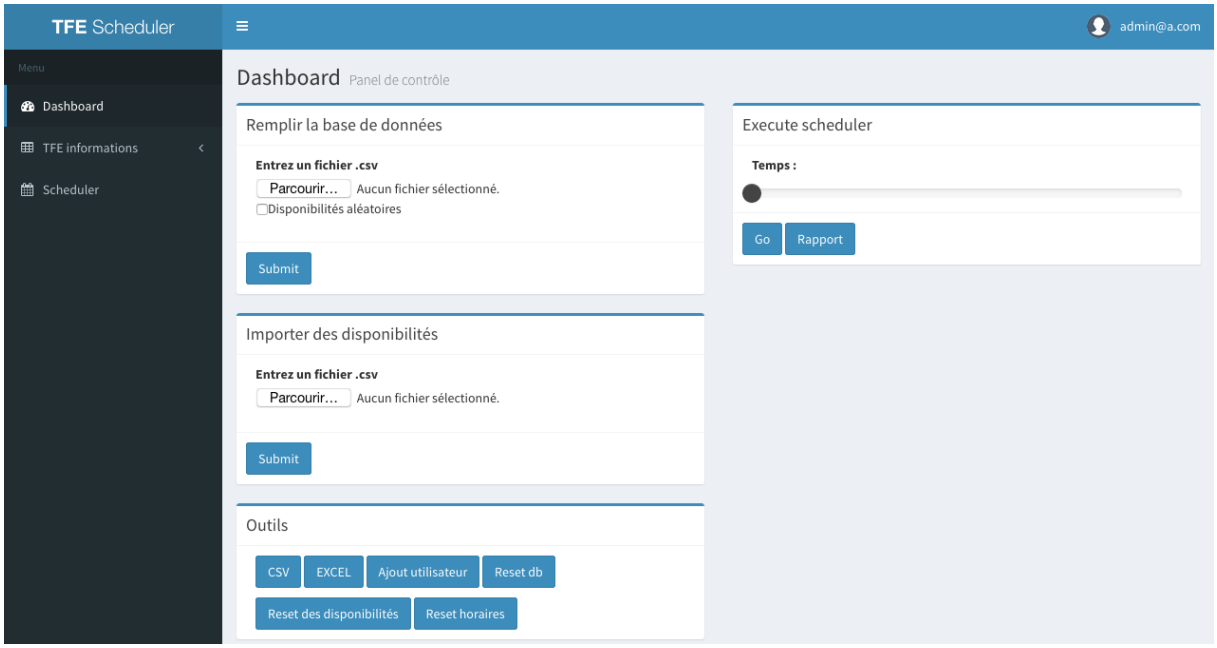


Figure 8.3: The admin index a.k.a. the dashboard

This page (figure 8.3) is the same page as the normal index page, except that an admin has full access to all the functionalities displayed.
As you can see, you can now populate the database with a .csv format coming from the application of Vincent Legat that is currently used by the students and the secretaries. This will able you to easily import all the data in one click. It should be done once at the beginning and you can also generate random availabilities for the jury (useful for the testing phase). These random availabilities are based on the availabilities of the previous academic year (2015-2016).

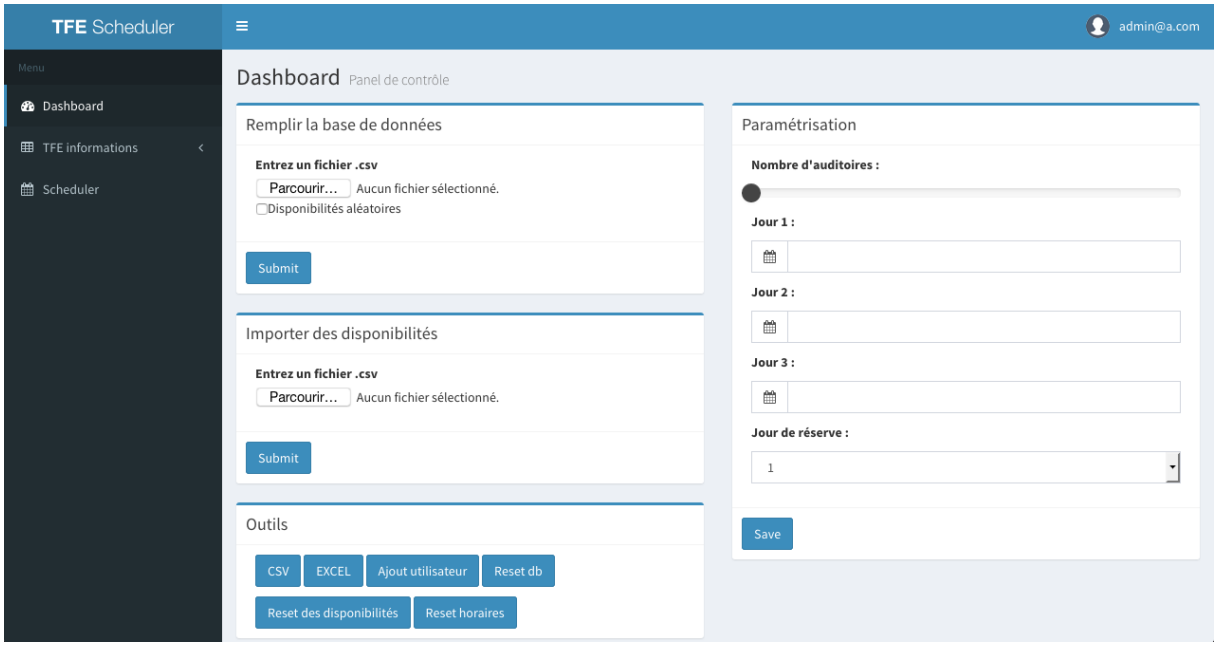


Figure 8.4: The admin index (parameterization)

When clicking on the "submit" button, the server receives the data and parse it into the database. Now you have access to the parametrization of the solver (figure 8.4) that you should see on the right of your screen. The parametrization allows you to choose the number of auditoriums assigned to the master thesis defenses, choose the three days of the schedule and say what day is the reserve day. As a reminder, the reserve day is a day where no TFE should be assigned.

Once the parametrization is saved, access is given to the scheduler page (It is not useful to display this page until the days have been specified) and to the execution of the solver. The user will specify the time wanted to allow the solver to compute the best solution and then runs the solver. As the best solution will be given at the end of the specified running time, it is advised to have it occurring during the night. This should be done at least once, but it can be done as many times as necessary to try to assign all TFE's without session. After that, a small report about the results of the solver is created. Clicking on the "rapport" button will allow the user to access the report generated. This report gives some statistics and the conflicts among the juries.

In order to meet each jury availabilities, the availabilities should be entered manually. It is understandable that it is a tedious job for secretaries. Therefore, importing availabilities from doodle (http://doodle.com) has been allowed. It is actually a .csv file containing, in the first column, the emails addresses of the persons participating in the TFE's. The next 12 columns represent the different time slots. Each of these 12 cells must contain **OK** (if the jury member is present) or nothing at all (if the jury member is not available). This last feature was realized by Mr Ludovic Taffin who should take over the project for the next academic year. He took care of this parsing of availabilities because this addition is a "nice to have" (a part of the program we could do without).

On this page, all the information that the database contains can be reset except for the user's information. It is useful to start from fresh when we have finished the tests. It is also useful when jumping from the June session to the September session.

With the same idea, the availabilities can be reset (say that all the juries are available for all the sessions) and so can be the schedule (say that all the TFE's are not assigned).

One last thing that can be done on this page is to add a new user. When clicking on the button "Ajout utilisateur" you will receive a message pops up asking to enter the email address, the password and the permission of the user (0 for a base user and 1 for an admin). Of course, for security reasons, the password is hashed in the database with the md5 algorithm in order not to be able to see it with a simple request.

8.2.4 TFE information page

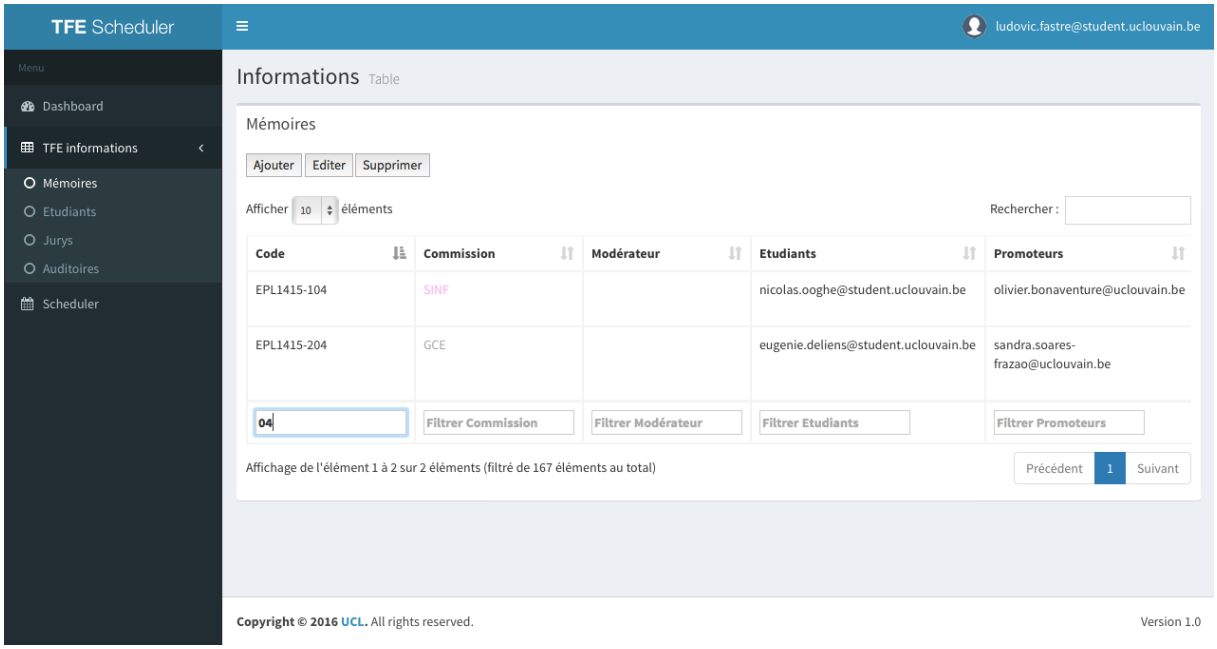


Figure 8.5: The TFE informations page

On this page (figure 8.5), all information about the different master thesis is displayed. In order to facilitate the reading of the information, access to a table containing the code, the commission, the moderator, the students, the advisors, the readers, the title and three parameters (confidential, cpme, openhub) is allowed.

Searching a data in this table is possible on the top right. This search is instantaneous, it will directly show the results.
The possibility to filter the data in the columns by entering what is wanted in the boxes below the table is also given.
By clicking on the label of the column to sort, the user will be able to display the information by alphabetical order.

Of course a new TFE can be created on this page by clicking on the "Ajouter" button. It will open a interface (modal) asking to enter the information.
Updating an existing TFE is also allowed : just by selecting it and clicking on the "Editer" button. It will open the same interface as for adding, will show the information of the TFE edited.
Students, promoters and readers are here for information purposes. They can not be modified on this page as the other information and are editable only on their allocated page.
The last thing authorized on this page is to remove a master thesis by selecting it and by clicking on the "Supprimer" button. This will not only remove the TFE, but will also delete all the data related and associated to it (see the section about the database 7). These relations are the links between the students and the TFE's for example.

8.2.5 Students information page

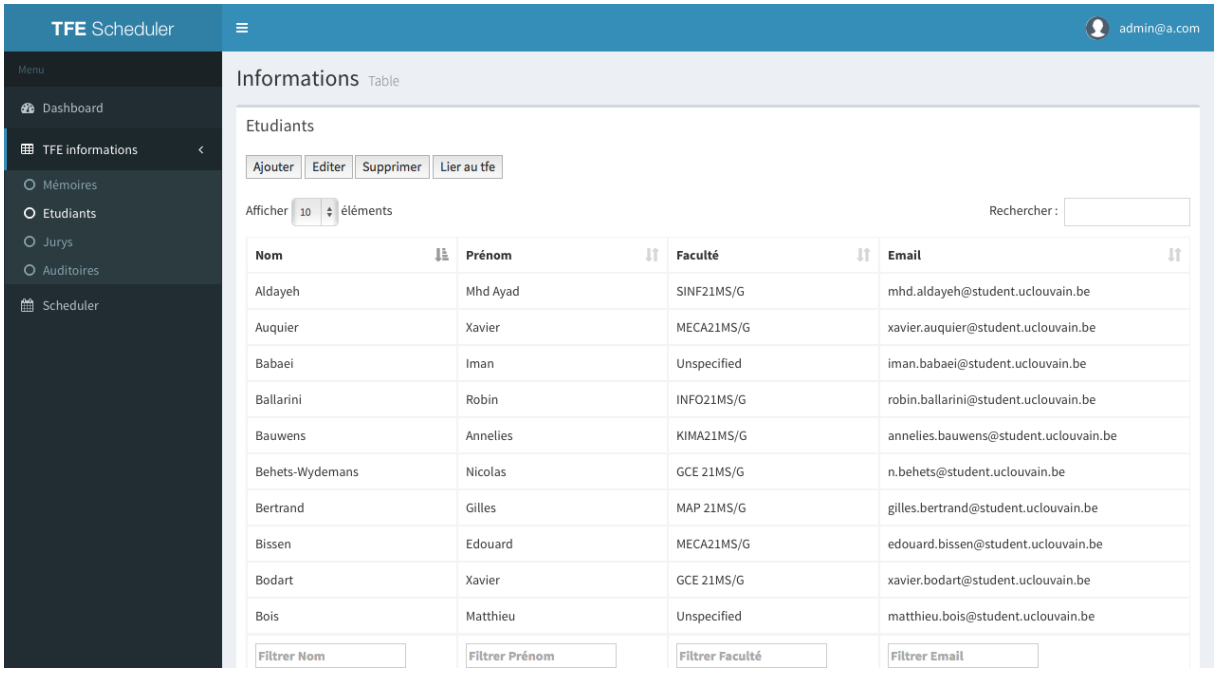


Figure 8.6: The students informations page

This page (figure 8.6) is another page displaying information. In this case, it concerns the students. As for the database, a student has a first name, a last name, a faculty, and an email address.

The search, the filters and the sort functionalities are the same as for the TFE page.

A student can be added by clicking on the "Ajouter" button. It will open an interface asking to enter all the information on the new student.
Parameters related to a student can of course be edited by selecting the student and clicking on the "Editer" button. It will open the same interface as for adding, but the information of the student you are editing is displayed.
The second last button ("Supprimer") allows the deletion of the student selected. By doing this, the relation between him/her and his/her master thesis is removed.
The last thing allowed is to link a selected student to a TFE by clicking on the "Lier au TFE" button and selecting a TFE from the list of existing master thesis. A student can only have one TFE.

8.2.6 Persons information page

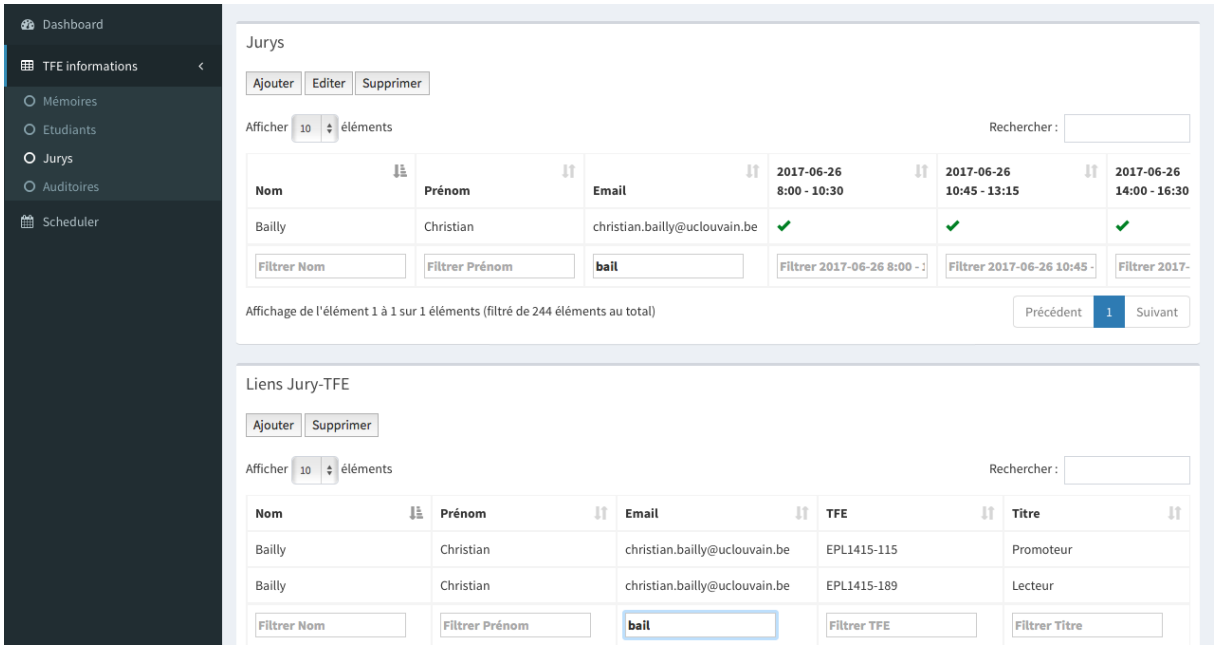


Figure 8.7: The persons informations page

This page (figure 8.7) is another page on information. In this case, it concerns the persons (juries). As in the database, a person has a first name, a last name, an email address and a list of availabilities associated to him/her.

The search, the filters and the sort functionalities are the same as for the TFE page.

The contents of this page are separated in two sections : a table for the juries and a table for the links existing between the juries and the TFE's.

In the first table you have all the information related to the jury except the TFE to which he/she is linked. On this table, a jury can be added by clicking on the "Ajouter" button. It will show you an interface asking to enter all the information about the new jury.
A jury can also be edited by selecting him/her and clicking on the "Editer" button. It will open the same interface as for adding, but will display the information of the jury being edited.
The last thing allowed to do on this page is to remove a jury by selecting it and clicking on the "Supprimer" button. By doing this, the relations between him/her and his/her master thesis is also removed.

In the second table, you have the name of the jury, his email address, the thesis to which he/she is linked and the title he/she has for this thesis (advisor or reader).
You can add a relation by clicking on the "Ajouter" button. It will show an interface asking to choose the jury, the title of the jury and the TFE. By doing this, you link a jury to a master thesis he/she has to attend.
You can also delete a link between a jury and a TFE by clicking on the "Supprimer" button.
As you can see, this system is different from the relation between a student and a TFE. It is so because a jury can be linked to multiple master thesis, something that the student is not allowed to be.

8.2.7 Auditoriums information page

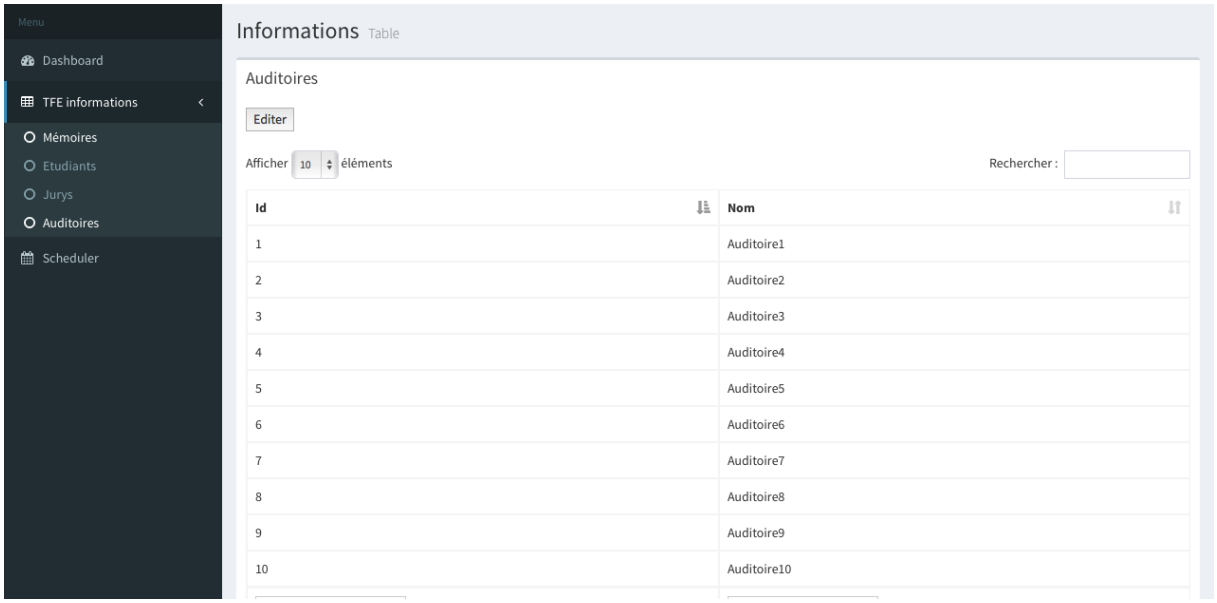


Figure 8.8: The auditoriums informations page

This page (figure 8.8) is the last one on information. In this case, it concerns the auditoriums (rooms). As in the database, an auditorium has just a name.

The search, the filters and the sort functionalities are the same as for the TFE page.

The auditorium are already created when you do the initial parametrization (specifying the number of rooms allocated for the schedule). So the only thing you can do on this page is to edit the name of the auditorium selected by clicking on the "Editer" button. It will show you an interface asking you to modify the name of the auditorium.

8.2.8 Scheduler page

Figure 8.9: The scheduler page

This page (figure 8.9) is the most important page of the application. This is where all changes to the schedule will be done.

After launching the solver (see admin index page), the results will be displayed as shown above. Each box corresponds to a master thesis represented by its code and its commission (a defined color).

On the top left of the screen, each commission is displayed with its associated color. It is for information purposes only as it allows to have a better understanding of the color codes.

Each unassigned TFE is found on the bottom left of the screen under the "Mémoires" label. Assigning a TFE to a session is done by dragging and dropping it on the scheduler.

The scheduler is displayed in the middle of the page. On its left, you will see the auditoriums (that you can modify on the auditorium page). There are as many auditoriums as specified in the initial parametrization. And on top of the table, the days and hours of the scheduler are shown (also specified in the parametrization).

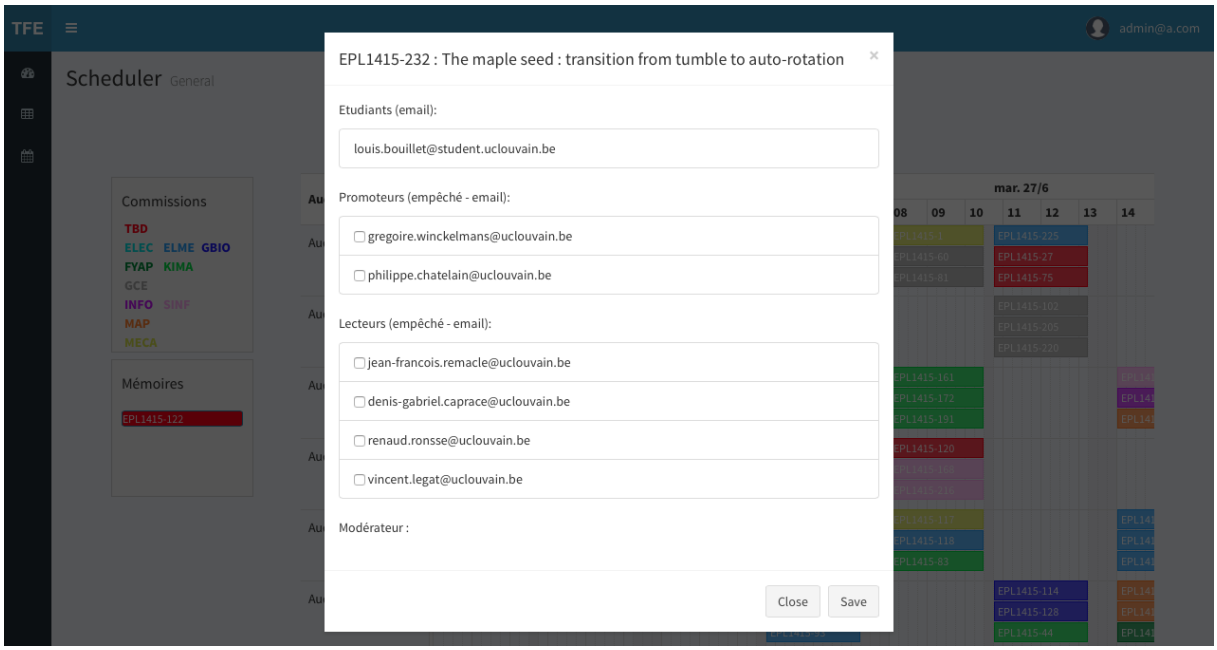


Figure 8.10: TFE details

By clicking on the TFE (figure 8.10), you will have access to all its participants (students, advisors, readers and moderator). This is also where you can set a member of the jury as "prevented". This will have the effect of being able to assign a master thesis even if its member of the jury is in conflict (for example in parallel with another session). Each member of the jury has a checkbox on the left of his/her name, that's where you want to set the "prevented" status. You can drag and drop each TFE in the scheduler, but also out of it (in the box on the left). If a jury member of the TFE you are trying to assign to has conflicts with the session, the application will show you the conflict(s) in question and your TFE will go back to where it was. This can be prevented if you set the status of the juries concerned as "prevented".

If you want to make changes to the master thesis information, it is recommended to open another tab (or window) with an information page at the same time.

Chapter 9

Concurrent access of the application

An important part of the application is the concurrent access to it. Each secretary must be able to access the application at the same time and be able to modify information at the same time. In this way, they can coordinate more easily and agree more quickly than by exchanging Excel files. The data to be affected by this concurrency is the TFE assignments to the sessions. This is the most important factor. We can not change a TFE session without everyone being informed.

First, in order to be able to use the database concurrently, it is necessary to use locks. Fortunately, a lock mechanism is built within SQLite so it was not needed to create it.

Then, each user must be identifiable in order to manage the concurrency. This is why a user table was created and used in conjunction with the Webpy session module. The session module provides session support and may contain useful information.

Finally, we need a way to manage this concurrency. The best way to do it would have been to create a schedule that changes instantaneously each time a user edits it. That means that it will allow everyone to see the changes made in real time. But this solution is a complex one and it would have required more time to be realized and more resources. The algorithm used for merging collaborative edits from multiple peers is called operational transformation and is not a trivial one. Furthermore, documentation is hard to find on that one.
Another solution was to block the schedule access when someone is using it. This solution is too restrictive because the secretaries could not work on the schedule at the same time.
So the best solution was to be able to define if the TFE you are moving has been modified without you being aware of it. If so, the program will simply notify you and refresh the page to view the changes. It was therefore necessary to add a timestamp field to the TFE keeping in memory the last time this TFE was modified. And the user would also have a timestamp associated with his session which updates each time it refreshes the page and see the schedule updated. It is therefore sufficient to compare the 2 timestamps to know if the user can modify the TFE (TFE timestamp $<$ user timestamp). The only disadvantage of this method is that the user often needs to refresh the page when changes to the same TFE's are done by other users at the same time. The user will also have to refresh the page if he/she modifies twice consecutively the same TFE because his timestamp is not modified while the TFE timestamp is. This last point can be solved by adding a field to the TFE to know what is the identity of the last user who modified it.

Chapter 10

Server Management

In order to be used by all the secretaries at the same time on the same data, the web tool needs to be deployed on a server. The INGI department was willing to lend one of its machines for the project and the server is therefore accessible only from the UCL network. The port 8080 is open and provides access to the website at `http://130.104.78.219:8080/`.

10.1 Installation

In order for the program to start, Python, Java, Webpy, Gurobi and Openpyxl must be installed on the server beforehand. Then downloading the sources at `https://github.com/Siceron/webTFEScheduler` or making a git clone of the repository. When it's done, you just need to run the command "python3 models.py" to create the database and "python3 adminLTE.py" to run the webpy server.

10.2 Backup

A script generating a daily backup of the database is available also. You can run it with the command "python3 backup.py" in another process. To do this, you can use screen [8] which is a multiplexer of terminals. You can then run the server and the backup script at the same time.

The backup allows, in case of crash or serious bug, to be able to recover former data. Fortunately it did not have to be used during the testing phase, but remains necessary.

The backup system was running during the production phase and saved the database everyday.

Chapter 11

Tests

11.1 Solver tests

These tests were done at the very beginning of the master thesis project development and were designed so that the solver is entirely independent. This means that it should not be taken into account that the data provided to the solver has already been tested in the web part.

These tests will mainly serve to verify that one modification in one part of the program does not alter any another one.

There are four categories of tests, one for each part of the program:

1. Verification of the accuracy, integrity of the data sent by the user
2. Check that the commission attribution works with this data
3. Check the corect run of the solver, its correct reply and its correct report creation
4. Verification that the JSON file output is correctly given

11.1.1 JSON parsing tests

Indeed, what is interesting to test is the behaviour of the program according to the inputs given by the user. So the solver tests focus mainly on the data that the user enters in the application via a JSON file.

The tests will verify that all the cases listed below return an exception :

- The user refers a json file that doesn't exist
- A field in the JSON file is missing
- A field inside an object is missing (eg: the code of a TFE is missing)
- The type of a variable is not the expected one
- A character is missing when defining a list or a JSON object (eg: "[" missing)

11.1.2 Commission distribution tests

These tests will make sure that the correct commissions are attributed to a TFE. For example, if two students are in SINF (sciences informatiques), the commission of the TFE should be SINF.

11.1.3 Scheduler tests

Here we are not interested to know if the program has found the most optimal solution, but whether a solution has been found. This is why the tests verify that a solution has been found and that a report has been generated.

11.1.4 Output tests

Here the program will simply check that a JSON file has been created for commissions distribution and for the scheduler results.

11.2 Web tool tests

This part is more user oriented, it has been decided to carry out these tests by humans. Given the timing, this choice allowed me to implement more features and to be able to react quickly enough to correct the bugs. Indeed, if I had spent more time on tests, we would be less likely to fall into unexpected behaviors as we will see later in this section. However, I would not have been able to implement the application as it is in time and use it for the June session. The fact of moving faster in within the functionalities allowed me to have a constant feedback and I was able to adapt the program to meet the users needs, requests and updates.

For the organization of these tests, I divided them into several phases corresponding to the different routine steps used by the secretaries for the organization of the master thesis defenses schedule.

- 03/04/2017 : Give access to people attending meetings. These people have tested all the features of the application to their liking and have reported the errors. This first phase of testing did not include the routine of the secretaries but was important to solve general bugs before giving the secretaries their accesses.
- 24/04/2017 : Encoding data in the application. In this phase, all the data on the TFE's was encoded. The majority of this data was processed by the parser which took the information from Vincent Legat's website. This part was really critical as we wanted to try to parse a maximum of information while being as generic as possible. This could lead to unexpected bugs.
- 26/04/2017 : Encoding availabilities in the application. This part should not cause bugs because the encoding was realized in the application. But with the new functionality of Ludovic Taffin, namely the parsing of availabilities from a doodle file, some bugs could have arosen as in the previous step.
- 02/05/2017 : Make the schedule. This is also a critical part because it was the first time that the solver was used with real data. This may have led to adapting the solver if it is not performing well enough.
- 08/05/2017 : End of the test phase, beginning of the production. At this stage, all tests were completed.

Now we are going to talk about the bugs I could detect and correct throughout the various tests on the application. Thanks to this I was able to have a correct phase of production in time.

11.2.1 First phase

As a reminder, in this phase, I tested in all directions without any particular organization. The aim was to find as many bugs as possible before involving the secretaries.

Since this phase was not really organized, I did not face any bugs. Therefore it was then transformed into teaching phase of the tool to the various stakeholders.

After discussion, Mr Glineur suggested to have a functionality to enter jury availabilities as for TFE's information. As explained above (see section 8.2.3), Ludovic Taffin took care of the realization of the latter.

11.2.2 Second phase

As a reminder, in this phase, tests concerning the encoding of data in the application were performed. So this part is mostly about the functionality parsing the data.

The first bug found was not really a bug because here the use of excel file generated a file with a special encoding. The parser can only read csv files with a correct UTF-8 format which is widely used everywhere, so the solution was to encode the file in UTF-8 format.

Then a second more concrete bug was found : the parser stopped after a certain number of TFE's read. The problem came from an external jury that had not been encoded in the same manner as is the staff of the ucl : with the last name and first name separated by a comma. This generated an error and it was necessary to add this particular case to the parser.

Then a third bug was found. The parser stopped again, but this time because of a description of a jury. Indeed, each member of the jury is separated by the sequence " - " that the parser uses to differentiate them. This time a member of the jury had a description between paranthesis comprising the sequence " - " inside it. We do not use the descriptions in the application, so we just had to disregard the parentheses for the juries and the problem has been resolved.

We found that some juries were present several times in the database after parsing these. This is not normal behavior of the system as each email address that represents a jury member is unique (this field is unique in the database). So it happened that these juries sometimes had a space in their emais address and were therefore considered as two different people. This problem was easily solved by adding a "trim" function to emails removing those spaces.

Finally, some TFE do not have a jury and the application simply ignored them. But Mr. Glineur said it was important to add them anyway and, so for these TFE's, we added generic juries (eg: missing@missing.com) to fill the gap.

In conclusion, we can observe that the bugs generated by the parsing of the data concern mainly particular cases coming from the application of Mr. Legat. But now these have all been dealt with.

11.2.3 Third phase

As a reminder, in this phase, we tested the encoding of availabilities in the application. As expected, we did not find any errors with this functionality.

But we have observed a particular case that has not been dealt with. Indeed, if a TFE is assigned to a session and we change the availability of the jury members of that master thesis, the TFE remains in the session even if it is in conflict. We have therefore taken the decision to withdraw all the conflicting TFE's from the session after the availability of one of the members has been changed.

A similar problem is the addition of a new member to the TFE jury. Indeed, the latter could have availabilities which are not in agreement with the current session of the master thesis. But it is also possible that the same person finds himself/herself in a parallel session of the TFE concerned. In both cases, we managed this problem in the same way as above : we withdrew all the conflicting TFE's from the session after the addition of a new jury member.

11.2.4 Fourth phase

As a reminder, in this phase, we tested the use of the automatic scheduler in the application. Since this part is tested separately (see section 11.1), we could not find any bugs but we were able to say if the solver needed to be adapted.

After a run of 8 hours on the actual data, the solver gave us rather disappointing results. In fact, only 46% of the TFE's were assigned. How was that possible ?
After reading the logs, we saw that the first model was unfeasible. Indeed, as explained in the section 6.4, it appeared that a jury member of a TFE was available less time than he/she should be due to the number of TFE's in which he participates. In this very case he was available for 1 session but he had 4 TFE's. The first model was infeasible because each TFE must be assigned but it was impossible.
Since this is a particular case, we thought it was better to simply remove all the relevant TFE from the variables of the model. As a result, we left the choice of the assignment of these TFE's to the secretaries.

11.2.5 Fifth phase

As a reminder, this phase is the production phase. At this time, all the tests and corrections were done. But it happens in reality that some unseen bugs appeared. These cases were very dangerous as we were already in the production phase and the data are supposed to be definitive. It was therefore very important to react as quickly as possible to correct program errors and sometimes to manually modify corrupted data in the database.

A bug such as this one appeared because, at one point, a web page was no longer accessible as it generated an internal error. In fact, the page (jury in this case) wanted to display jury-TFE relations or the TFE in question no longer existed in the database. It turns out that when removing a TFE, the relations related to that TFE were not deleted. This omission has therefore been taken into account and relations without TFE have been removed manually.

The last bug and probably the most serious one was that some jury members found themselves on parallel sessions without having the "prevented" status. This error happened because a parameter in the detection of jury in parallel remained hardcoded. Only the first 5 audiences were monitored. But the damage was done and several jury members found themselves in parallel while they could not. I therefore especially created a tool to find these TFE's in conflict and

to be able to report them to Mrs. Poncin who handled them with the help of the solver.

After these last modifications, the schedules were ready to be published.

Chapter 12

Improvements

This chapter will focus on improvements that could be made to the application. During development and meetings, several ideas were listed and considered "nice to have" because other features were more important. Furthermore, this application was carried out over the 2016-2017 academic year on a limited time, so some points still need to be improved, especially from the point of view of the user's comfort. The improvements to be added concerning the solver and the web application will be, therefore, listed here under :

12.1 Solver improvements

- The solver should take into account the prevented status of a jury for a TFE that has not yet been assigned. This must be effective during the first run and during subsequent runs.

- The number of TFE's of one commission per parallel session should be reduced. For the same time slot : 1-3 TFE's is acceptable, 4-6 TFE's should be avoided and 7+ TFE's should not be allowed. Indeed, it is complicated or impossible for a secretary to deal with 7+ TFE's distributed in 3 different auditoriums.

- The number of sessions per day should no longer be fixed at 4. Some days may only have sessions in the morning (as in September when there are fewer master thesis defenses).

- It would be necessary for the solver to group together the TFE's with the same commission in the same auditoriums.

12.2 Web tool improvements

12.2.1 Scheduler interface

- Moving the TFE's more easily into the cells. It is especially important to improve the detection of displacement.

- There should be a better synchronization system. A system that allows to see the changes in real time without having to refresh the page.

- It should be possible to filter the schedule by commission/local/jury (in table or graph form).

- When clicking on a TFE, one should be able to observe the list of time slots where all juries are available. And among these time slots, show which ones are available (not having 3 TFE's).

- A bubble should be added for each TFE displaying additional information (e.g: student) when the cursor hover on it.

- It should be possible to define the order of the TFE's within a session.

12.2.2 Other functionalities

- There should be a view to switch a complete session (with 1 to 3 TFE's) from one auditorium to another.

- It should be possible to export and import a schedule in order to try changes and be able to go back to an old version without having to go through the backup.

- It should be possible to extract the schedule in order to broadcast it (in pdf format or any other one) or create a public webpage to show it.

- Each TFE should have a category field in order to be able to filter them. This way, it should be possible to assign a TFE to a future schedule instead of deleting it (e.g: assign it to September 2017).

- There should be a log of the movements that have been made to the TFE's including the date, the actor (including the solver) and a description of the movement. This log should be accessible to all users.

- Parametrization should be more flexible. The user should be able to define each session individually (with parameters like date, time, reserve, ...).

Chapter 13

User's feedback

Before going on to the conclusion, a very important point in achieving a product is to have feedback on it. This will help determining whether the program is easy to handle, whether it is appropriate for users, whether there are improvements to be made, and so on.

The secretaries were then confronted with a series of open questions on the application developed in order to better target their feelings and needs. The following is an overview of these questions and the resulting overall opinion :

What do you think about the interface of the website? Do the proposed views match your needs ? Should there be others and if so which one and for what use ?
To this question, it was answered globally that the interface was quite intuitive and met most expectations. This is especially due to the fact that their remarks were taken into account. On the other hand, they said it would have been interesting to have an extra detailed view for the schedule (which could be used for publication).

What problems did you encounter when using the tool?
All agreed that there was no problem except that the area to drop a TFE in the scheduler view is a little bit too small but that they got used to it.

Did the automatic schedule creation help you ? If so, to what extent ?
Yes, it allowed to have a first result facilitating the work. Previously, some secretaries did not use the availabilities because they estimated that the jury members should be available all the time. The tool helped them doing it and so the juries should be happier now.

Did you use the extraction of data as excel/csv? If so, why ?
No secretary really used this feature, or just out of curiosity.
But the Vice-Dean, Pr Glineur, used it a lot. Especially for testing purposes.

What are the advantages of this program compared to the old method ?
Overall, the advantages remembered were:

- An overview of all the TFE's of the various EPL commissions.
- A first result thanks to the solver that can be modified later.
- Messages warning about the different conflicts.
- Taking into account the availabilities.

What are the disadvantages of this program compared to the old method ?
Overall, the disadvantages remembered were:

- The fact that all commissions are mixed up.
- The fact that the jury members concerned by a specific TFE are not shown directly in the scheduler view.

What improvements would you like to make to the program ?
Several ideas for improving the program were given:

- Exporting the view of the scheduler
- Choose colors more suitable for commission (some characters are hard to read because of this)
- Add availabilities for students

Would you be willing to use this application for future master thesis defenses ?
In the end, all the secretaries interviewed agree to say that they want to reuse the application.

In conclusion, we can say from this interview that users are generally satisfied with the application, that it is better than the old method, but that there are still some improvements to be made.

Chapter 14

Conclusion

From now on, secretaries no longer need to exchange hundreds of emails to organize the master thesis defenses schedule. They have at their disposal a tool that allows to create a schedule in agreement with the other commissions to which thesis are related. Furthermore, the solver developed enables them to benefit of a first automatically generated proposal. This proposal, as it has been demonstrated, allows them all kind of modifications (move, cancellation, addition of TFE, etc.) while still respecting the constraints imposed by the availabilities of the jury members. So, in the end, all the stakeholders implied in the planning exercise of the master thesis defenses will look at a schedule including all the EPL's Program Commissions and, of course, respecting the availabilities of each member of the jury.

This experience has been very rewarding both at the apprenticeship and at the professional level. In fact, when I started this master thesis, I was not very familiar with web development. Using new tools and technologies has taught me a lot, like for example the limitations and sometimes the interactions between Webpy, SQLObject and Gurobi. I am confident that this first approach of these technologies will help me a lot in my professional life.

Even if I was working alone on this thesis, I realized that organization is a primordial issue. It is thanks to this organization that I was able to submit this report in time and to have a first prototype of the program that could be used for this session (the June 2017 session).

I also learned that, when developing such a tool or any other kind of program, a developer must remain as generic as possible when implementing functionalities. Indeed the client can change its mind at some point (at any time during the course of the project). Therefore, being generic makes it easier to adapt to any situation.

This thesis also allowed me to put into practice what I learned during my years of studies at UCL and see that it is rewarding. For example, having learned how to use an Agile method allowed me to put it into practice and helped me to better organize the project. Also, when I had to take care of the solver, having decided to follow an option in artificial intelligence allowed me to better analyze the optimization problem : that means to define a model with variables, an objective function and constraints. Or again, having a course in database gave me enough basis to elaborate a correct model following some good practices like using atomic fields (no lists) and making a relational database. All this experience helped me to deliver a program that meets the expectations of the customer.

I also perfected the use of some technologies I already knew like, for example, Python or SQLite. Using them in a real problem with a client and not in a theoretical project only was clearly different. The software doesn't only require to give a good result, but it also requires to

be maintainable to able to modify it in the future. Furthermore, as the user interacts sometimes in unexpected ways with the program, it is really important to cover all the cases that might appear and not only the ones that will give the expected result.

Facilitating the work of others is important to me. The feedback of the secretaries has greatly encouraged me and confirmed the choice of my future profession.

I appreciated the opportunity to be in front of a customer. Its exigences in term of utilisation allowed me develop a tool more efficient and adapted to the needs of the secretaries in the specific context of organization of master thesis defenses schedule. It is of an extreme importance to listen to the final users of the software. To listen to their needs, their expectations and their wishes.

We have seen that the tool could be improved in several ways and it might be interesting to test other models or other technologies to solve the problem in a more optimal way. But compared to the old method, we have already made a big step forward.

At the end of the journey, even if the tool could be improved, the client was satisfied with the product I delivered to them and it helped them to save a lot of time in the process of making the master thesis defenses schedule. The proof of it is that the schedule of this master thesis defense session was organized highly in time with the developed tool. That mean that the goal has been reached.

Chapter 15

Annexes

15.1 JSON input template

```
{
  "reserveDay": int,
  "sessionDays": int,
  "sessionRooms": int,
  "advisors":
  [
    {
      "email": String,
      "faculty": String,
      "disponibilities": bool[]
    },
    {
      "email": String,
      "faculty": String,
      "disponibilities": bool[]
    }
  ],
  "readers":
  [
    {
      "email": String,
      "faculty": String,
      "disponibilities": bool[]
    },
    {
      "email": String,
      "faculty": String,
      "disponibilities": bool[]
    }
  ],
  "tfes": [
    {
      "code": String,
      "students":
      [
        {
```

```
            "email": String,
            "faculty": String
          }
        ],
        "advisors":
        [
          String,
          String
        ],
        "readers":
        [
          String,
          String
        ]
      }
  ],
  "fixed":
  [
    {
      "code": String,
      "session": int
    }
  ],
  "banned": [
    {
      "code": String,
      "session": int
    }
  ]
}
```

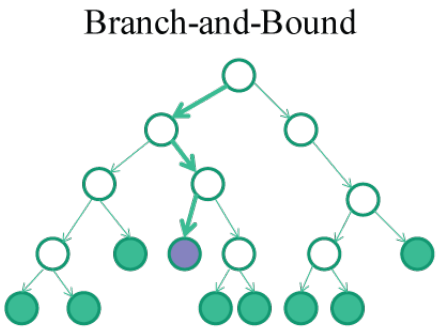
15.2 JSON output template

```
[
  {
    "code": String,
    "session": int
  }
]
```

15.3 Gurobi optimizer

So, how does the Gurobi optimizer work ?
Basically, it is generally solved using a linear-programming based branch-and-bound algorithm. The algorithm will first remove all of the integrality restrictions (some/all variable must take integer values). Now we have a linear-programming relaxation of the original MIP. If it happens that the values respect the integrality restrictions, then this solution is an optimal solution of the original MIP. But it doesn't happen very often. So now we have a variable, say x, who have a fractional value. For example, if the value of x is 1.4 in the LP relaxation, we can exclude this

value by imposing x $\leq$ 1 and x $\geq$ 2. And x is now called a branching variable.
Now we have two MIP's, one where x $\leq$ 1 is imposed and one where x $\geq$ 2 is imposed. This way, we have replaced the original problem by two problems more restricted. In our case, the algorithm will separate the variables equals to 1 or 0 since all the variables are between 0 and 1. And we can continue the same reasoning with the two new MIP's. By doing that, we generate a search tree. And of course, if we reach a point where all the constraints are satisfied, we have a solution.



Each node in branch-and-bound is a new MIP

Since we have a minimization problem, we can update the upperbound each time we reach a better solution than the last one. We also have also a lower bound obtained by taking the minimum of the optimal objective values of all of the current leaf nodes. With that, we can obtain the gap that is the difference between the upperbound and the lowerbound.

Gurobi perform also a presolve algorithm that reduce the linear problem. Basically, it will combine the constraints. Following their example, suppose we have the following constraints :

$$x_1 + x_2 + x_3 \geq 15 \tag{15.1}$$

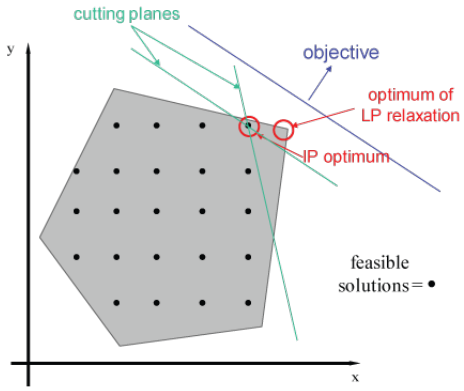
$$x_1 \leq 7 \tag{15.2}$$

$$x_2 \leq 3 \tag{15.3}$$

$$x_3 \geq 5 \tag{15.4}$$

As you can see, the constraints can only be satisfied if $x_1 = 7$, $x_2 = 3$, and $x_3 = 5$. In this case, we can remove the variables and their constraints from the formulation. This will have a great impact on the overall size of the problem.

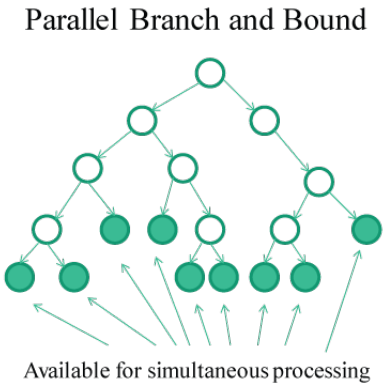
The optimizer can also perform a technique called cutting planes that tighten the formulation by removing undesirable fractional solutions.



Cutting Planes

As you can see with the figure, the idea behind that is adding new constraints in order to exclude undesirable solutions. The optimizer will only add these constraints if it knows they will help. By that, it will have a great beneficial effect on the solution process.

Another good point of the optimizer is that the solver runs in parallels. Since every node of the search tree is a different MIP problem, it can be processed independently.



15.4 Functionalities

I now will list all the functionalities by date of realization. I know that this is only a exhaustive list and that the application can be ameliorated.

- 02/02/17 : Create a database model with SQLObject containing the TFE's, students, advisors and readers (Very important). We need to store the data for everything, especially for the automatic scheduling.
- 02/02/17 : Send a csv file, parse and store all its data (Very important). The csv file come from the application of Vincent Legat. The file contains the code, the title, the students, the advisors, the readers of the TFE and the emails of the students, advisors and readers.
- 02/02/17 : Create a page to show the current data : TFE's, students, advisors, readers (Very important). This is important because we want to have visibility on the data and in the future be able to modify them.
- 04/02/17 : Get the informations from the solver and store it (Very important). Here we get the proposition of sessions distribution from the solver.

- 04/02/17 : Get a view of the scheduler (Very important). This will be the key tool and we want to have a pretty view of the result coming from the solver.
- 05/02/17 : Create a JSON containing the data and execute the solver with the JSON as input (Very important). In order to communicate with the solver (.jar) we have to send the informations with a comprehensible format : JSON.
- 05/02/17 : Show details of a TFE in the scheduler view (Important). Seeing only code of the TFE's isn't really representative, that's why we are able to see their title, advisors, readers, students by simply clicking on them.
- 05/02/17 : Same availabilities for same advisor and reader (Very important). If the advisor and the reader are the same, it is important that they have the same availabilities.
- 08/02/17 : Add parameters to the solver (Important). Like the number of auditoriums and the maximum time allowed running the solver.
- 12/02/17 : Be able to set the sessions in the scheduler view (Very important). That's nice to have a pretty schedule, but we need to be able to modify it.
- 12/02/17 : Be able fix sessions for the solver in the scheduler view (Very important). If we want to re-launch the solver, we want to keep some TFE's fixed before doing it.
- 12/02/17 : Show the report of the execution of the solver (Very important). Some nice informations about the execution of the solver : the time, the sessions allocated, the number of TFE's impossible to assign (availabilities of the persons never match), the ratio of TFE assigned and all the conflicts.
- 18/02/17 : Create user accounts that can access to the application (Very important). At this state, everyone could use the application, but now it is restricted and this will be used in the future for a concurrent utilisation of the application.
- 19/02/17 : Make the tables editable in their respective view (Very important).
- 20/02/17 : Merge readers and advisors table into one : persons (Important). It was bad to use two tables and have duplicated values, so we merged it into one.
- 20/02/17 : Add the possibility to link students/persons to their TFE (Very important). Now you can add remove or modify the relation between a TFE and a student/person.
- 21/02/17 : Collect the commissions of the TFE from the solver (Very important). This is the other part of the solver, assign the commissions in order to know who is in charge of this TFE.
- 21/02/17 : Add color for the commissions (Very important). In order to be more readable, I added colors for each commission. With that, you can easily differentiate the TFE's in the scheduler view.
- 25/02/17 : Ask a confirmation when you want to move a TFE in the scheduler (Important). This is to avoid to assign a TFE by accident to an unwanted session.
- 25/02/17 : Concurrent access to the TFE's (Very Important). With this, the secretaries will be able to work at the same time on the schedule. The system is pretty simple : when you enter the page, the session register the current time. Each TFE has also a timestamp that updates when they are modified. So basically, to know if the TFE is synchronized, it computes the difference between the two timestamps. If the TFE has been modified prior entering the page, it's fine. In the other case, it will simply refresh the page and thus update the timestamp of the session.

- 27/02/17 : Show conflicts when you move a TFE to a new session (Very important). If the scheduler tool don't let you move a TFE, it is important to know why.
- 09/03/17 : Add a parametrization to the solver (Very important). Be able to select the days for the scheduler and the number of auditoriums.
- 10/03/17 : Possibility to add a reserve day in the parametrization (Very important). The reserve day is a day that we should avoid when we assign the TFE's.
- 18/03/17 : Possibility to add a moderator for the TFE (Nice to have).
- 21/03/17 : Export the data in csv and excel format (Important). This is nice if you want to use the informations in excel or an other program.
- 22/03/17 : Add more details and filters to the tables (Nice to have). This change is meant to be comfortable for the user.
- 23/03/17 : Add the possibility to modify the name of the auditoriums (Nice to have).
- 24/03/17 : Possibility to add the prevented status to a person (Important). The prevented status is added to the relation between a person and the TFE. This status means that we don't take into account the conflicts about the person. This allows us to assign a TFE conflicted.
- 24/03/17 : Show the TFE's that are in conflicts (Nice to have). This is also for the comfort of the user. The conflicted TFE's are annotated with a small "*" next to their code.
- 24/03/17 : Differentiate an admin and a common account (Nice to have). In this case, a simple user will not be able to launch the automatic scheduling or modify the parametrization.
- 30/03/17 : Add fields to the TFE (Nice to have). Theses are the confidential, cpme and openhub fields. They are not used by the applications, but it is simply an information coming from the application of Vincent Legat that we don't want to lose.
- 31/03/17 : Possibility to add users in the admin panel (Nice to have).
- 31/03/17 : Add a backup system for the database (Nice to have). It is important to have the security to have a backup if the server crashes or the data are corrupted. A backup is programmed every day at 3 a.m.
- 31/03/17 : Possibility to reset the database in the admin panel (Nice to have). If we want to start from fresh, this button will remove all the data except the users.

Bibliography

[1] Abdullah Almsaeed. AdminLTE website. `https://adminlte.io/`. Accessed: 11-04-2017.

[2] Ian Bicking. SQLObject website. `http://www.sqlobject.org`. Accessed: 06-04-2017.

[3] Inc. Gurobi Optimization. Gurobi website. `http://www.gurobi.com`. Accessed: 06-04-2017.

[4] D. Richard Hipp. SQLite website. `https://sqlite.org/`. Accessed: 06-04-2017.

[5] Google Inc. Gson repository. `https://github.com/google/gson`. Accessed: 11-04-2017.

[6] FullCalendar LLC. Fullcalendar website. `http://www.sqlobject.org`. Accessed: 11-04-2017.

[7] SpryMedia Ltd. DataTables website. `https://datatables.net/`. Accessed: 11-04-2017.

[8] GNU Project. Screen user's manual. `https://www.gnu.org/software/screen/manual/screen.html`. Accessed: 23-05-2017.

[9] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[10] Aaron Swartz. Webpy website. `http://webpy.org`. Accessed: 06-04-2017.

[11] Bootstrap Core Team. Bootsrap website. `http://getbootstrap.com/`. Accessed: 11-04-2017.

[12] Laurence A Wolsey. Mixed integer programming. *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**